UiO : Department of Informatics
University of Oslo

Use of Artificial Intelligence and Machine Learning algorithms to predict programming levels in Cochlear Implant patients

Master thesis

Andreas Høyer Iversen

November 2014

2014

Use of Artificial Intelligence and Machine Learning algorithms to predict programming levels in Cochlear Implant patients

Andreas Høyer Iversen

http://www.duo.uio.no/

Trykk: Reprosentralen, Universitetet i Oslo

Abstract

It is today possible to partly replace the hearing of people with profound sensorial hearing loss, using a Cochlear Implant (CI). The CI is a surgically implanted device, which has the purpose to replace normal acoustic hearing with electrical stimulation of the auditory nerve fibres, which gives the user a hearing sensation. A CI is therefore able to partly restore hearing where normal hearing aids cannot. The surgery is only one part of the challenging process to make the CI work properly, after the surgery there is a period where the CI is programmed to fit the patient. The programming is done by setting a number of parameters. There is today no known way to use data collected from earlier patients to set these parameters accurately, therefore how these parameters are set and how well greatly depends on the audiologist and the cooperation of the patient. Small children are especially challenging, since they are rarely able to give reliable feedback.

In this thesis I investigate possible ways a computer utilizing machine learning can improve this process. While the ideal goal would be a program that can adjust the parameters all on its own, a program that can give suggestions to the range of possible values could also help a lot. To make these predictions the program can use post- and intraoperative measures from other patients that have successfully received a CI. There has been a dataset with about 300 patients available for this project. Using the data from these patients it has been investigated if it is possible to use an algorithm to find some pattern in the data that can be used to make CI parameter predictions.

The results show that using the data collected from earlier patients (ESRT, ECAP, impedance and patient age), it seems impossible to make prediction accurate enough for an automatic fitting process. However, the result show that using the patient database it is possible to achieve an accuracy that can still be useful as a starting point for a programming session.

The results also show that if it is possible to first find a few of the CI parameters, a model can be used to make predictions in order to find the rest of the parameters with fairly high accuracy. During the experiments the optimal electrodes for interpolation and prediction was also found. Which electrodes being chosen seemed to have a large impact on the model performance, and this was independent of the model used. This means that even if a prediction model as the one proposed in this thesis is not used, it is still beneficial to use the found optimal electrodes when using e.g. linear interpolation. However, using the model proposed in this thesis gives lower prediction error than linear interpolation which is sometimes used in practise today.

Preface

I would like to thank my supervisors Ralf Greisiger, Professor Jim Tørresen and Associate professor Ole Jacob Elle. I would like to express my gratitude for their time spent on giving me great feedback and encouraging attitude. Without their continuous support and invaluable input that made me keep on working when nothing seemed to go as planned, this thesis would not have been possible.

I would also like to thank all the people who have contributed to the research that this thesis has used, references to these sources have been provided where possible. This thesis would not have been possible if it was not for all the great work of others.

Finally, I would like to thank family and friends for always being helpful and supportive.

Contents

List of Figures

1 Introduction

1.1 Motivation

According to the WHO there are more than 360 million people worldwide with loss of hearing [1]. Depending on the cause of deafness many of these can benefit from hearing aids that amplify sound waves. There are however some cases that are too severe for normal hearing aid to be of help. In these cases it is possible to bypass the damaged parts of the inner ear in order to stimulate the auditory nerve fibres directly. The auditory nerve fibres are responsible for sending sounds to the brain and by stimulating them directly a hearing sensation can be achieved. This hearing sensation is different from normal hearing, and the patient needs some time to learn and adapt to the new way of perceiving sounds. Depending on various factors, such as the duration of deafness and reason of deafness, the performance with a CI can vary a lot, but many people are able to hear good enough to follow a normal conversation and in the best cases the hearing and language skills are close to those with normal hearing, at least in quiet listening situations.

The process of getting a CI requires the implant to be surgically implanted behind the ear. The surgery is however only one part of the process, after the surgery a sequence of sessions with an audiologist will follow. During these sessions the audiologist tries to adjust a number of parameters of the speech processor to fit the patient as good as possible. The aim is to let the patient hear as good as possible without discomfort. In order to set these parameters the audiologists use the data found during and after the surgery, this is commonly referred to as *objective data*. Since the objective data is often not sufficient to make an accurate setting, feedback from the patient is also used; this is commonly referred to subjective information. Sometimes it is difficult to get any useful feedback from the patient for various reasons. Young children are especially challenging, since they do not give accurate feedback on how well they perceive sounds. There is also a risk of giving too much current/loudness, which can be uncomfortable for the child and my lead the child to refusing be the device. It is also important not to set the parameters too low, because then they may hear sounds too weak. This is especially important for young children since hearing is important for how well they learn speech and language at early age. Studies have shown that the earlier a child gets the implant the closer they can come to the auditory skill of children with normal hearing [2].

Today, there exists no objective measure based method to program the CI, meaning that the programming is done mainly based on feedback from the patient, combined with what experience the audiologist have. This makes it especially difficult for audiologists that are new or unexperienced with programming a CI. Unexperienced audiologist may need additional time to find an accurate fitting, which is problematic because the patient has just a limited duration of concentration during CI programming. The longer the programming takes, the more likely it is that the patient starts giving unreliable feedback. Since the feedback is

important for how the parameters are set, an increase in time it takes to find the correct parameters may have a negative impact on the final result or require more programming sessions.

There are many other areas that have benefited from using a machine learning model that can make accurate predictions based on patient data [3-6]. Using a prediction model to predict the CI parameters could potentially automat the programming session, or reduce the programming time. If the programming took shorter time it would be beneficial for the patient, but also reduce the total cost of getting the CI to work. The reduced programing time may improve speech perception scores, especially for young children where the period of giving reliable feedback can be very short, whereas infants almost never give reliable feedback. Even for adults that are post-lingual deafened[1], it can be difficult to give accurate estimate of loudness. This makes the programming specially challenging for clinics and audiologists with little experience in programming Cochlear Implants, and they might have problems doing an accurate programming and providing adequate sound. In these sorts of situations a model that can assist the programming could be very helpful.

1.2 Problem Formulation

During the surgery of the CI a set of measurements are performed (objective data), these measurements are done to check if the device functions properly and used to predict what levels of current the CI should use when stimulating the auditory nerve fibres. Measurements can also be performed postoperative while the patient is awake. All this data is saved to a database, together with programming parameters found by the audiologist.

This thesis investigates if it is possible to create a program that can be used to guide CI programming for new patients using data from previously programmed patients. The program will be based on a machine learning model, which is created using the patient database. The idea is to use information about patients that have successfully received a CI and see if it is possible to find patterns in the data that can tell us something about future patients. The ideal solution would be a program that can find parameters for the CI without any help from an audiologist.

Creating a system that automatically finds every parameter with a very good accuracy seem improbable, since there are many variables that affect the outcome, e.g. reason of deafness, duration of deafness, mental state, social environment, etc. It is more plausible that the system can give advice like starting points or a range of possible values. In order to give programming advice the program can use objective data about patients together with the correct CI parameters found by audiologists on previous patients.

[1] Post-lingual deafness is deafness that occurs after the development of speech and language.

1.3 Limitations

While the goal is to investigate the possibility to create a program that can give advice for programming the CI, there will not be enough time to actually try out how well the program works in practice. The model will however be tested on a separate test set to get an estimate for how accurate the model might be, and since this is data from real patients it should give a fairly good estimate. It will also not be enough time to create a complete program that is suited for practical use. This thesis will mainly focus on investigate if creating such a program seems possible and what kind of assistance it may be able to provide.

1.4 Structure of the thesis

This thesis contains six chapters. In chapter 2 I present some background about hearing and cochlear implants. Chapter 3 takes on some theoretical machine learning background that will be used in chapter 5. Chapter 4 is about the patient database, where I use statistical methods to analyse the data and discuss what sort of possibilities this data may give and what limitations it might have. In chapter 5 I present my results and some of the experiments that were performed to explore what possible uses the data may have. In chapter 6 I discuss some of the results and what potential application the results from this thesis may have, followed by some thoughts on possible further research.

2 Cochlear Implant background

This chapter goes through some background knowledge related to Cochlear implants. First there are some background related to hearing and deafness (2.1). Then, some information about how a Cochlea implant works (2.2) and what kind of procedure and adjustments are required for the implant to function properly (2.3). The last section explains some of the patient related data (2.4).

2.1 Hearing

Sound

"Sound may be defined in terms of either psychological or physical phenomena. In the psychological sense, a sound is an auditory experience—the act of hearing something. In the physical sense, sound is a series of disturbances of molecules within, and propagated through, an elastic medium such as air." From [7] (p31)

This definition takes on the two different ways to look at sound. One is the physical definition, where sound is a chain of air molecules pushing and pulling on each other. The other is the definition that sound actually is a psychological phenomenon and is simply the brains interpretation of signals traveling along the auditory nerve. These signals are in normal hearing created by our ear that has the ability to transform acoustic energy into electrical signals, much like a microphone do.

Sound has various attributes that our ear can distinguish; two of these are sound frequency and amplitude. Frequency is the distance between each wave, higher frequency sound is perceived as more high pitched. How high the pressure is in each wave is the sound amplitude, higher amplitude is perceived as louder.

2.1.1 Normal hearing

The first part of the hearing process is that sound reaches the pinna, which is the visible part of the ear. The pinna collects sound such that it enters the auditory canal and hits the eardrum (see Figure 1). The eardrum is a cone-shaped piece of skin and it has the same role as diaphragm in a microphone. The sound waves cause movements of the eardrum and it is very sensitive so even a small change in pressure will move it. If the sound is perceived as very loud a protection mechanism triggers which causes the tensor tympani muscle and the stapedius muscle to contract, these muscles are connected to the eardrum in such a way that when they contract they stiffens the eardrum and decreases the amount of sound passing through, this especially effects the lower frequencies [8]. Connected to the eardrum is the malleus, vibrations in the eardrum cause the malleus to move back and forth. The malleus is

connected to the incus, which is again connected to the stapes; these three bones are called the ossicles. Movements of these bones will cause the stapes to push on the cochlea fluid through the oval window (see Figure 1).

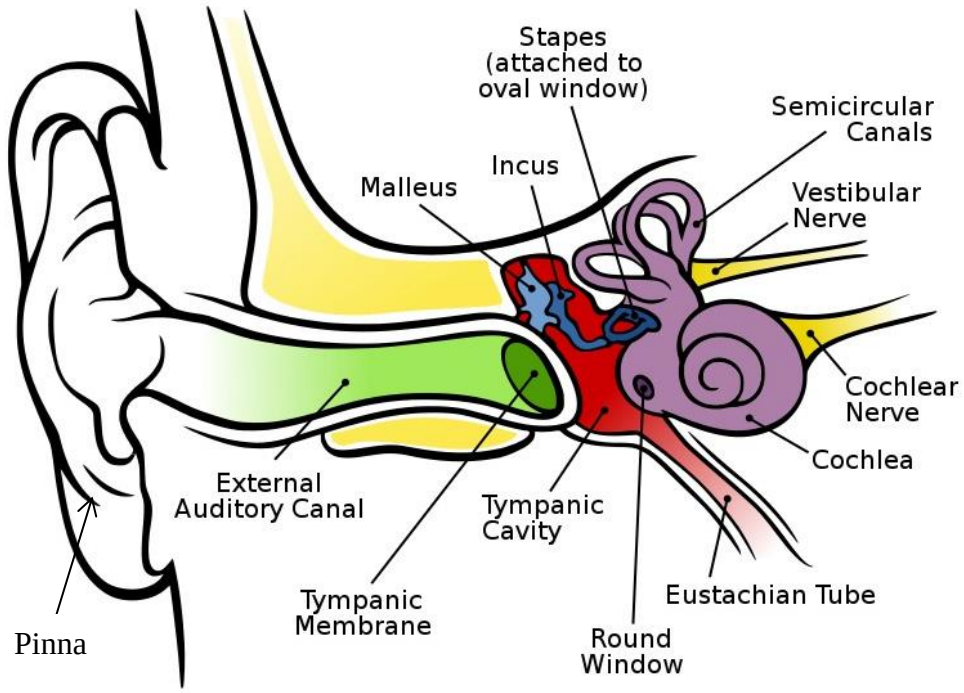

Figure 1– The anatomy of a human ear (from [9])

The cochlea has a snail formed structure, with a bony shell and canals with membranes containing fluids inside. When the stapes moves back and forth it will cause a travelling wave along the basilar membrane. On top of the basilar membrane is the organ of corti (see Figure 3), which contains the hair cells - this is where the transductions happens from a movement to an electrical impulse. The hair cells are connected to the cochlear nerve and the movement in the basilar membrane will cause these cells to send electrical impulses along the cochlea nerve (see Figure 3). These nerves are connected via the auditory brainstem to the brain trough the cerebral cortex where the signals get interpreted as sound.

The various sections of the membrane have different attributes, so that different hair cells are triggered to different sounds. The start of the cochlea is tuned for the high frequency, up to 20 000Hz; while the end of it corresponds to low frequencies, down to 20Hz. Louder sound will cause more movements in the fluid which will cause more hair cells to be activated. The

brain can then determine the frequency and loudness according to activated hair cells.

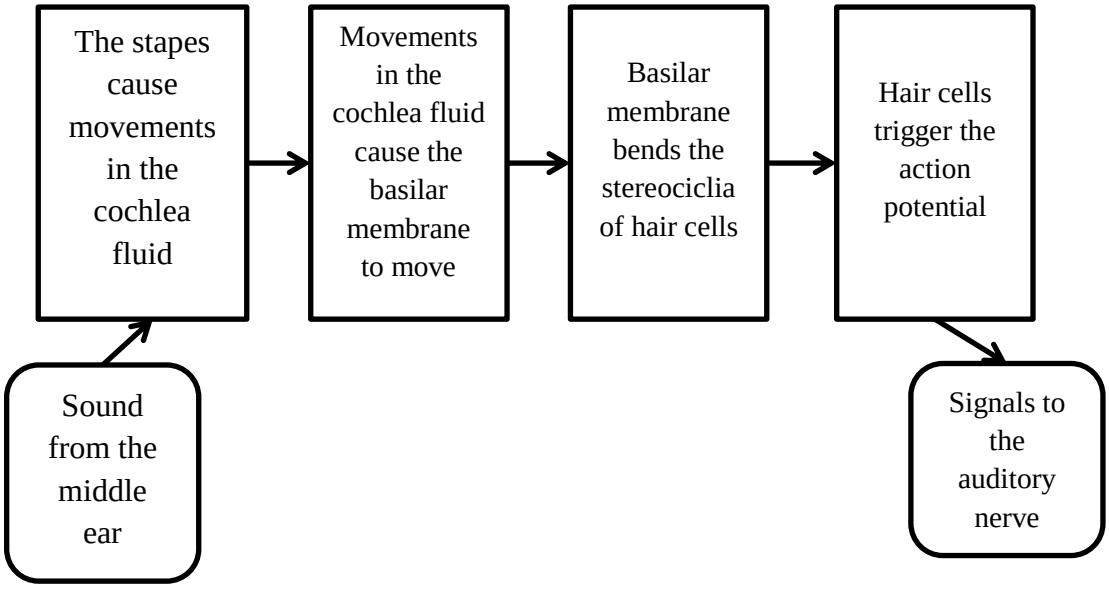

Figure 2– Flow chart of the cochlea functions

There are two types of hair cells, inner hair cell and outer hair cell. Only the inner hair cell works as sensory receptors. The role of the outer hair cells is to reinforce/sharpen the signal in order to enhance frequency discrimination [10]. There are about 12, 000 outer hair cells and 3,500 inner hair cells. On top of each hair cell is the stereociclia. Movements in basilar membrane will make the stereociclia deflect, which triggers a stimulation of nerves cells. This triggers an increase in electrical activity in the neurons; this is called the *action potential*. The action potential propagates along the auditory nerve to the auditory cortex in the brain, where the signals are interpreted as sound.

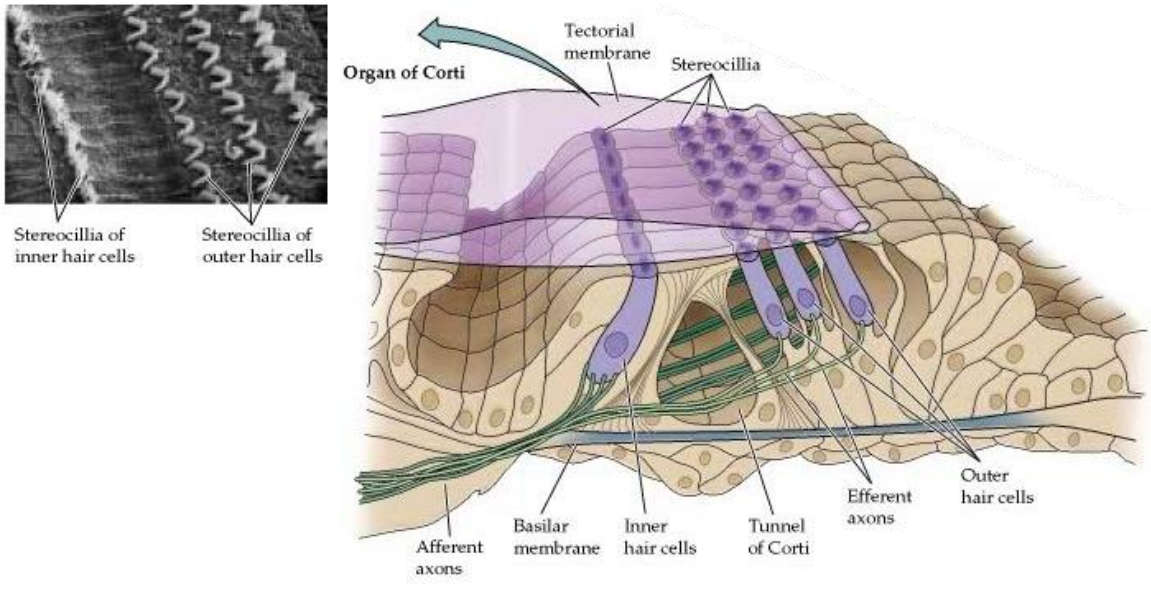

Figure 3- Shows a magnification of the organ of Corti, from [5]

2.1.2 Deafness

There are two main categories of hearing loss, conductive hearing loss and sensorineural hearing loss. Conductive hearing loss means that the sound waves do not reach the cochlea for some reason. This can be due to damage on the ossicles, an infection can have caused fluids to build up and caused permanent damaged or something can have happened to the eardrum, like a perforation. Sensorineural hearing loss is caused by a dysfunction of the inner ear, the auditory nerve fibres or the auditory nerve. The hair cells in the cochlea are the most sensitive parts of the hearing system if they are damaged or partly damaged the patient will have a sensorial hearing loss. If only the outer hair cells are damaged this limits what is perceived by the inner hair cells, but they are still able to perceive sounds above about 40 to 60 dB [11]. Common causes for deafness are infections, exposure to loud noise, genetic, ototoxic[2], head trauma or birth complications.

There are multiple levels of hearing loss; it can vary from moderate, severe to profound hearing loss. Moderate to severe hearing less means some hearing, but it implies difficulties in hearing conversational speech. Profound hearing loss, means very little or no hearing. If hearing loss is moderate to severe it is possible to improve the hearing with hearing aids, by amplifying the sound. The improved loudness can help and hearing aids can work with just a small amount of hair cells, but if the hair cells are completely damaged it is impossible to amplify. In this case the cochlear implant can help, because it stimulates the cochlea nerve fibres directly, bypassing damaged/not functioning hair cells and electrically stimulate the nerve fibres.

2.2 The Cochlear Implant device

2.2.1 How a Cochlear Implant works

A CI consists of two parts, an external part that consists of a microphone and a speech/sound processor. The external part uses a microphone to pick up sounds, which is used to create a set of stimuli the electrode array[12]. The processed and digitalized information is sent to the internal part (the implant). The implant is located behind the ear under the skin. It receives signals from the external part and converts the signals into electric pulses which it transmits along the electrode array. The electrode array is inserted in a way such that each electrode stimulates one part each of the cochlea (see Figure 4). As previously explained (in 2.1.1), the various parts of the auditory nerve corresponds to different perceived sound frequency, this is exploited by the CI such that it can choose what sound the user hear. It is also possible to change the perceived loudness of the sound by giving more or less current, higher current will lead to higher perceived sound, since this will activate more nerve fibres.

[2] Ototoxic means something that is toxic to the ear. This is often a medical induced drug like antibiotics.

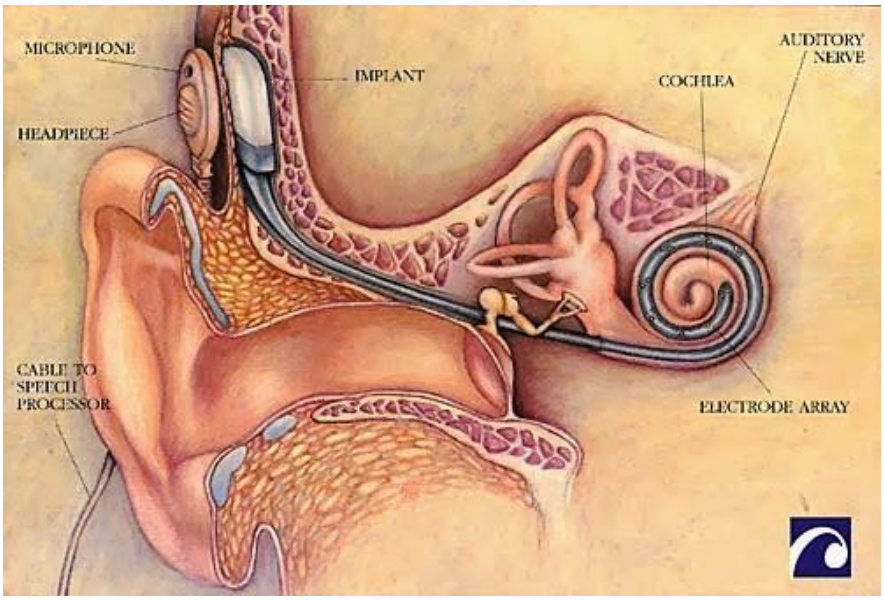

Figure 4 -- Shows a cross section of the ear with an implanted CI

The number of electrodes varies, but the CI used in the data set available for this thesis has 22 electrodes. In contrast normal hearing has a few thousand hair cells (see section 2.1.1), which make it easy to imagine that hearing with a CI cannot be as good as normal hearing. When sound is perceived in normal hearing it is a combination of all these hair cells that are used, because the CI only have 12 to 22 electrodes (depending on device), it has a much lower resolution.

2.2.2 The electrode Insertion

The CI needs to be surgically inserted behind the ear. The surgeon drills a whole through the bone behind the ear; this gives access to the middle and inner ear. The surgeon also makes a small hallow spot in the bone for the implant housing, also behind the ear underneath the skin, see Figure 4. The electrode is then inserted through an opening in the cochlea near or through the round window, using hand precision. About 4-6 weeks after surgery the implant will be turned on for the first time and the process of programming the device starts.

2.2.3 Programming the CI

The CI has a large number of parameters that can be changed in order to fit the patient. These parameters are important for how well the implant will work. Because of this a lot of time is spent trying to find the parameters that fit each patient. The configuration is done by an audiologist in multiple sessions after the surgery. The sessions are often a few weeks apart for the patient to get used to the new configuration. This means that the sessions often spawn over many months or until a satisfactory result is found.

There are a large number of parameters that can be changed, but many of the sound processing settings do not normally need to be changed and can be left with its default value. Some of the parameters can be adjusted to improve hearing in difficult sound environments, like noise level suppression. The T-and C-levels are the most important parameters that need to be set, they control how much current the electrode array use when stimulating the nerve fibres. What levels that fit the patient vary between patients and using the wrong values can reduce hearing performance or cause discomfort. T-levels and C-levels also change over time, which means the patient usually accepts louder levels as time goes by, because it takes time to get used to the new way of hearing. The parameters often change a lot from first session to the second, but levels normally stabilize after about 6 to 12 months [13]. This is why fine tuning of the parameters often take up to one year.

One of the main challenges when programming a CI is that there are so many electrodes that need to be programmed individually. Since there are many electrodes and each electrode have multiple parameters this can take a long time. What makes it especially challenging is that the programming rely on patient feedback, but the patient often loose concentration after a while and starts to give inaccurate answers.

2.3 Cochlear Implant parameters

There are multiple parameters that can be adjusted as explained in 2.2.3. Since there are mainly T-levels, C-level and the pulse width that are relevant to this thesis, only them will be explained further.

2.3.1 T- and C-level

The Maximum comfort level (C-level) and the threshold level (T-level) are two of the parameters that needs to be found during the programming process after the surgery. The C-level is the maximum level of current on an electrode that does not lead to discomfort. This corresponds to the loudest sound that the patient can perceive without discomfort. The threshold level (T-level), is in contrast to C-level the minimum current for a patient to be able to hear anything; it represents the lower bound for what current to use, when stimulating the nerve fibres (see Figure 5).

These values are found by trying out various level of current, and see how the patient responds. For C-levels it is undesirable with too high level of current since it can be painful and a too low current can limit how well the patient can hear. When finding T-levels the audiologist tries out various levels of current, when the patent can barely hear anything (the threshold), this is the T-level. The same process is performed for young children and infants, but because the feedback they can give at such a young age is very limited, the T-level is merely a measure of the level of current that cause any change in behaviour. C-levels are even

harder to find for children and infants, since it is very important that the child do not experience any pain, since this can lead the child to reject the device. It is therefore important that C-levels are set very carefully, but this is also why C-levels suffer from great inaccuracy because of the missing feedback and the fear of causing pain to the child.

The programming procedure is repeated multiple times after the surgery, how many depends on the needs. It can take several years until an optimal fitting or satisfactory result is achieved. It is very important for small children that they perceive sounds as good as possible, because of the maturation of the auditory cortex. The faster they get to a good hearing quality the better the auditory cortex develops speech understanding. There is a critical phase of maturation until approximately 7 years [14]. Children beyond that age are usually not able to develop speech understanding with a CI if born deaf, while early implanted children before an age of about 12 months hear almost as good as children with normal hearing, at least in speech testing situations [2, 15].

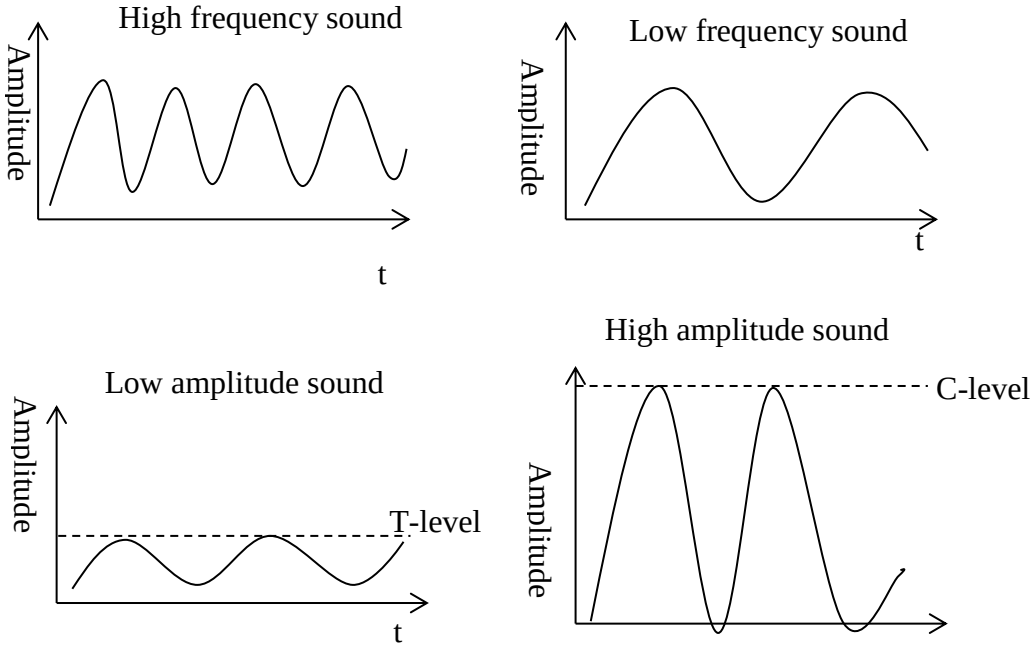

Figure 5 - Four different sound attributes. The “High frequency sound” is sound that would correspond to the area of the cochlea that is closest to the ear drum and high pitched sounds. The “Low frequency sound” correspond to the area of the cochlea that furthest away from the ear drum and low pitched sound. The “Low amplitude sound” correspond the softest sounds perceived and the T-level. The “High amplitude sound” correspond the highest sound that is still not uncomfortable and the C-level.

2.3.2 Pulse width

The pulse width parameter decides for how long a signal in the electrode array is held. This is different from the amplitude but gives a similar effect, because the longer the pulse is held the

more current passes through the electrode. Since the nerves fibres are sensitive to the amount of current, increasing the pulse with can give a similar effect as increasing the level of current [16]. One of the reasons for increasing the pulse width is in cases where the patient requires more current than the CI is able to provide. Among other things this can be caused by higher impedance on an electrode. By increasing the pulse width the CI is able to give more current using a bit more time, but this gives the same effect on the hearing experience. Since the important factor is the total amount of current that passes through the nerves fibres, we can use the formula $Q = I * t$ to express the relationship between the total current that passes through the nerve fibres given a current I and a pulse width t. Using this formula we can change I and t and still keep a constant Q, see Figure 6.

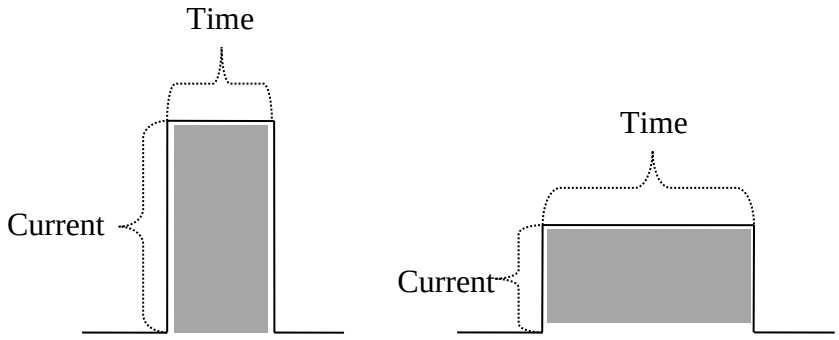

Figure 6 - Two signals with different pulse width and pulse amplitude, but the total charge (the grey area) is the same for both. Since the total current determines whether the hair cells trigger, these pulses will be perceived as equal for a patient.

2.4 The objective data

Since T-level and C-level are based on subjective behavioural observations, this can cause some inaccuracy. To complement the subjective data, it is desirable to use objective measures as well during the programming process. Therefore a set of objective measurements is performed during surgery and after. The idea is that these measurements can say something about what levels to use without having to rely on feedback from the patient.

2.4.1 Electrically evoked compound action potential (ECAP)

Electrically evoked compound action potential (ECAP) is a measure of the response from the auditory nerve fibres to electrical stimulation. How the action potential is triggered under normal hearing was explained in 2.1.1, but now electrical stimuli are used to measure the threshold for when the action potential is triggered. The measure is done using one electrode

to stimulate the nerve and use a neighbouring electrode to measure the response. The response is then transmitted to an external recording system. The ECAP has been proposed as an indicator for C- and T-level by numerous studies [17, 18]. In our dataset there has been found poor correlations between ECAP and C-/T-levels (this is discussed further in 4.2.2).

Relationship between hearing quality and the ECAP on CI users with short electrode array has been investigated [19, 20]. However the results have generally shown poor correlation. The ECAP has also been used as an estimator for C-levels, but with limited success [21]. While others have found a positive correlation between ECAP and C-/T-levels for infants [22]. There seems to be some variations for how useful various clinics find these parameters which may come from variations between the CIs or clinical practises.

2.4.2 Impedance

Before the device is used for stimulation it is tested for faulty electrodes using impedance measurements. The impedance is found for each electrode, it is measured between two electrode contacts. In which case an electrode has high or low impedance it might be faulty and should be turned off. High impedance can mean that there is something wrong with the implant itself, but it can also mean that the conditions around the electrode are causing poor conductivity. This can happen if the cochlea is bony or dried out[23]. If an electrode has low impedance it could mean that two electrodes have short circuited. The impedance also says something about the maximum level of current that can be delivered from the current source to an electrode contact (the compliance level).

2.4.3 Electrically evoked stapedius reflex threshold (ESRT)

Electrically evoked stapedius reflex threshold (ESRT), is a measure of what level of current is required for the stapedius muscle to contract. The value is found by stimulating the cochlea nerve fibres with increasing current, until the stapedius muscle contracts. As explained in 2.1.1, this is a reflex that normally occurs when exposed to high and intensive sound. It is therefore reasonable to assume that ESRT could be correlated with C-levels.

In 2012 there was a multicentre investigation on ESRT and ECAP relation to C- and T-levels [13]. They had 117 patients from 14 centres all with a CI. They found that ECAP could be used to create an initial flat program. For patients that were born deaf they found that ESRT could be used to derive a more precise model, but patient feedback was still much more useful.

J. H. J. Allum, R. Greisiger and R. Probst [17], investigated the relationship between ESRT and C-levels. They found some correlation between ESRT and C-level, but the correlation depended on electrode number. Some of the correlations were very low (< 0.2), while others where better (> 0.6), but still not good enough to accurately estimate C-levels. Possible reason

the correlation was not higher is that while ESRT is a brainstem reflex mechanism, C-levels is dependent on how loud the patient actually experience sound.

3 Machine learning background

This section will first give a brief introduction to some machine learning concepts (3.1), followed by some theoretical insight into machine learning (3.2). Then there will be a short description of some popular algorithms for creating prediction models (3.3). The last section will discuss some more practical aspects of machine learning (3.4).

3.1 Types of machine learning

Machine learning is a branch within AI that is used for data exploration and data prediction. There are mainly three kinds of machine learning algorithms: reinforcement learning, unsupervised learning and supervised learning. Reinforcement learning learns some function by getting feedback on its performance without getting the actual answer, this is often used when the problem consists of multiple intermediate problems, but we only know what the final result should be. Unsupervised learning is more about exploring data and finding patterns and clusters in unstructured data. Unsupervised learning is often used on large amounts of data that we want to learn something from, like finding trends or patterns. In unsupervised learning there is no value we want to predict and no labelled data beforehand. The job of the algorithm is to come up with these labels by itself. In supervised learning some labels are normally known and we want to see if it is possible to predict unknown labels, in cases where only the data is available. Using large amounts of data the algorithm finds patterns in the data that can be used to predict the correct labels. Which one of these methods is used depends on what problem we want to solve. For the rest of this chapter, the main focus will be on supervised learning since that is what will be used in the experiments in Chapter 0.

3.1.1 Supervised learning algorithms

The general setting in supervised learning is that there are some known data pairs (x, y), where the x values are some observed data related to y. The value y is the value that we are interested in predicting, it is also called the target value. By using some machine learning algorithm we want to see if we can find some function h that can use x to find y. Depending on what y is we call it either a classification problem or a regression problem. For classification problems y is normally a category or some nominal variable. In regression the output can be any real number within some range. Both problems use some sample data (x, y) in order to train a model for predicting the target value with x, see Figure 7.

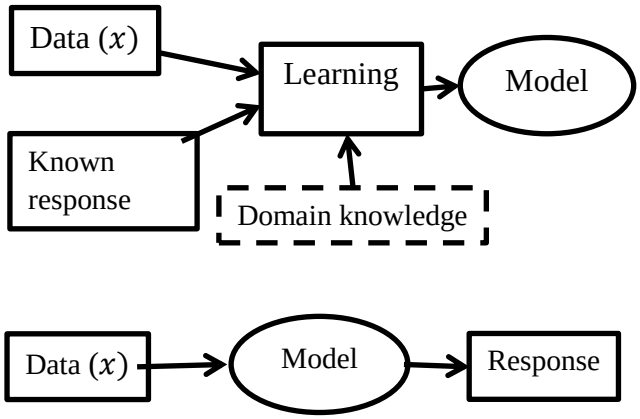

Figure 7 - A general overview for training a model. First a model is created from some learning algorithm with input data and the corresponding response as input. After the model is created the model can be used to find responses for new and previously unseen data. In order to improve performance if the models it is common to use some domain knowledge to choose algorithm or set parameters.

3.1.2 Classification and regression

There are many ways to look at a modelling problem; it all depends on what sort of information is required and what accuracy. Some problems require a simple binary classification, where the goal is to say if some data belongs to one class or the other. One example of this would be a model that can say if something is an apple or not. It is also possible to have multiclass classification problems where the model needs to find a most probable class among multiple classes. An example of this could be a model that looks at an image of a fruit and then tells what kind of fruit it is. A regression model is quite similar in many ways except that it is normally used to predict an arbitrary range of values. If compared to the previous example where a classification model would say what sort of fruit something is, a regression model could e.g. tell the size of the fruit or the weight based on some relevant data. Regression models can also be seen as a function approximation model, meaning that it tries to approximate some unknown function based on the observed data points.

3.2 Machine learning foundation

The previous section have explained that a machine learning problem consist of some data x that we want to use to find some target y, using a function h. The big question is how do we find this unknown function $h(x)$? One method to find $h(x)$ is called the maximum likelihood hypothesis, which states that the most likely function h is the function that maximizes the chance of seeing our data D, given that h is the true function. The variable D is defined as all observed pairs of data x and y. The most likely hypothesis can then be defined as $h_{ML} =$

$\text{argmax}_h \Pr(D|h)$, where $\Pr(D|h)$ is the probability for seeing data D, given the function h. The problem with this formulation is that we need to look at every possible function h, to find the one that maximize $\Pr(D|h)$. This is very impractical since it is infinitely many.

3.2.1 Deriving sum of squared error from Bayes rule

When using machine learning or some other statistical methods to predict the target values based on some known data, we cannot always expect to get perfect predictions. We do however want the most likely function given the data; this is known as maximum-likelihood estimation (MLE). In other words if we have some data X and the corresponding correct output Y, we want to use this data to find the most likely Y_i, when getting some new data X_i. Since looking at all possible hypotheses is infeasible a more practical method is required, and it turns out that Bayes rule can be used to derive a rule that is much easier to use in practice[24].

Noise in data is quite common, this means that if we have some true function $f(X_i)$ that maps X_i to the correct Y_i, the target value can be defined as $\overline{Y}_i = f(X_i) + \in_i$, where $\in_i$ is the error term defined as $\in_i = N(0, \sigma^2)$. This means that our data X_i with corresponding output $\overline{Y}$ is influenced by some Gaussian noise, with zero mean. Notice that we assume only Y is affected by noise, not X.

Given the data we want to find the most likely hypothesis, or the hypothesis that is the most likely to be the true underlying function $f(X)$. The most likely hypothesis is defined as: $argmax\ P(h \mid D)$, meaning we want to find the hypothesis h that is the most likely given our evidence/data X. If we assume uniform prior probability for all hypotheses this is the same as $argmax\ P(D \mid h)$. This means we want to find the hypothesis that maximizes the likelihood that we see our data. This can also be written as $argmax\ \prod_i p(Y_i|h)$, meaning we maximize the product over the probability for each of the data elements. We can write this as a Gaussian function as follows:

$$\prod_i argmax\ \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-\frac{1}{2}(y_i - h(x_i))^2}{\sigma^2}}$$

This formula expresses the probability that our hypothesis is the correct function $f(X)$, given that we have Gaussian noise added to Y and we have a data point X_i. This function can be simplified significantly and ends up as, $\text{argmax} - \sum(Y_i - h(X_i))^2$, which is the same as $\text{argmin} \sum(Y_i - h(X_i))^2$. Meaning that to get the most probable hypothesis we need to find the hypothesis with the smallest sum of squared error, where the error is the difference between the predicted value and the correct value.

3.2.2 The problems with sum of squared error

As mentioned there are a few assumptions that need to hold true for minimizing the sum of squared error to give the most probable hypothesis, but in real life problems they may not hold. We have assumed that Y (the value we want to predict) is affected by Gaussian noise with zero mean, while this may often be a reasonable assumption it is easy to find examples where this is not true. We can imagine that we want to use information about what a population eat to predict their weight. We ask them about what they eat and then we measure their weight. This is done in parallel with 2 different devices, one of them have a weight bias of -5, which means the weight is 5 less than it should be for half the population. This is not Gaussian noise and if a model was created using this data, minimizing the sum of squared error will not give the most likely hypothesis.

Another problem with our noise assumption is that $\overline{Y}_i$ is defined as $f(X_i)$ corrupted by some noise. The problem is we do not say anything about the noise in X, meaning we assume the input to be noise free. This is also not a certain assumption, if we relate this to the example above, we can imagine that people lie about how much they eat. Depending on the sort of noise and the magnitude we can run into some large problems when training a model. Dealing with these sorts of problems is not easy, but section 3.3.4 and 3.4.3 explains some algorithms that decrease the impact of such problems.

3.3 Some examples of learning algorithms

Depending on how much data is available and how good the data is, a supervised learning algorithm can learn very complex functions with very low prediction error. Supervised learning methods often use a try and fail method that consists of trying to predict a value, comparing the predicted value to the correct value, followed by correcting the error made. With this method the algorithm tries out a large number of hypothesizes always improving itself and hopefully in the end it will have a good hypothesis that fits the data well enough (or a low sum of squared error). How it improves itself depends on what sort of algorithm is used and there exist a wide variety of algorithms with different pros and cons. This section explains some commonly used algorithms for creating prediction models, some of which will be used for the experiments in Chapter 0.

3.3.1 Linear regression

Linear regression is a method that tries to find a linear function that fit some data, by minimizing the sum of squared error. Linear regression is used to find linear models, which means that the model is only able to use linear relationships between the input data and the target value when making predictions. While this may seem like a large limitation at first, it turns out that this is a property that can often be very useful. Since linear regression is very

simple both to implement and it is quick to run, it is often used before more complicated models, to see if a linear model actually is enough. Since the model is only able to find linear relationships it is very resistant to noise data and is able to find robust models, this will be explained in more detail in 3.4.2. Also if the data comes from some non-linear function the data can sometimes be transformed to fit a linear model.

Since a linear regression model is able to find linear relationships in high dimensions, it can find functions that are not always obvious to spot for humans. However, if the data and the target value have low correlations the linear models is unlikely to give good performance. Finding a way to transform the data to fit a linear model is not always feasible or possible, which is why linear models may not perform sufficiently when applied to complex problems.

3.3.2 Artificial neural network (ANN)

ANN's can unlike linear regression in theory learn any function[25]. An ANN is a popular algorithm partly because it is very generic and can be trained for a wide variety of tasks, including pattern recognition, classification and non-linear function approximation[26]. The idea behind ANN is to simulate how neurons work in our brain. In our brain the neurons are interconnected by dendrites and axons that let the neurons activate each other. To simulate this on a computer we represent the connections as values in a matrix, where each value represent the strength of the connection between two neurons. The network is able to learn functions by changing these weights in the matrix.

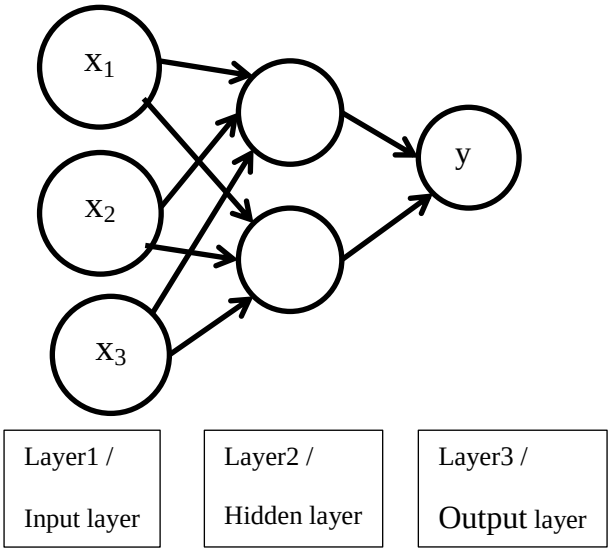

Figure 8 - An example of an artificial neural network. The network has 3 inputs: x_1, x_2 and x_3 these weights are connected to a hidden layer with 2 neurons, which is connected to the output neuron.

When the ANN trains it use the weights to predict labels or outputs, the output is then compared to the correct output in order to calculate the error. The error is used to change the

weights in such a way that next time the error will be smaller, this is done iteratively until convergence. There are many possible ways to update these weights; some methods simply use the gradient of the error as following: $w_k = w_k - \alpha * g_k$. Here the weight is updated as the previous weight minus the gradient of the error scaled with a leaning rate. This is the simplest way to update the weights and there exist a wide variety of other ways to update the weights[27, 28]. The cost function of a neural network can be defined as follows:

$$J(\theta) = -\frac{1}{m}\left(\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^i \log h_\theta(x^i) + (1-y^i)\log\left(1-h_\theta(x^i)\right)\right) \Bigg\} \text{Error cost}$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(\theta_{ij}^l)^2 \Bigg\} Cost\ of\ large\ parameters$$

The cost functions is a way to describe the performance of the model, meaning that finding a way to minimize this function will also minimize the error of the neural network. Generally we can use any numerical method for minimizing this functions or even genetic algorithms [29]. Depending on the number of parameters/neurons a neural network can learn very complex functions. Since a neural network is so easy to adapt to more complex problem, it is very applicable to a wide variety of problems. Section 3.3.1 explained that linear regression models are robust to noise since they are limited to linear functions. Since neural networks are not limited in the same way, it is possible for a neural network to fit noise. The second part of the cost function above is to reduce the complexity of the network and thereby stop it from fitting noise; this will be discussed further in 3.4.2.

Neural network training function

The fact that the neural network cost function can be expressed in such a way lets us use any method for minimizing the function. A popular algorithm to minimize such functions is the Levenberg-Marquardt backpropagation algorithm, this algorithm has proven to converge fast and give equally accurate models as other training functions [30]. The Levenberg-Marquardt algorithm (LMA) was first described as a numerical algorithm for minimizing non-linear functions [31]. It was later modified into an algorithm for training neural networks [32]. One of the big advantages with LMA is that it can find proper step sizes in every direction of the error surface[3]. If the training function uses a fixed step size it can give slow convergence if the error magnitude is not the same in all directions, see Figure 9. LMA uses the second derivative to evaluate the shape of the error surface which makes it possible to take an appropriate distance in each direction. If however the current area of the error surface is very complex the algorithm uses something similar to steepest decent algorithm [33]. With LMA method there is no need to set a learning rate as in many other methods, since it finds the step

[3] Error surface means a surface that gives the model error at any given point in the weight/parameter space.

size on its own. However, it requires that the cost function is the sum of squared error such as the cost function described earlier.

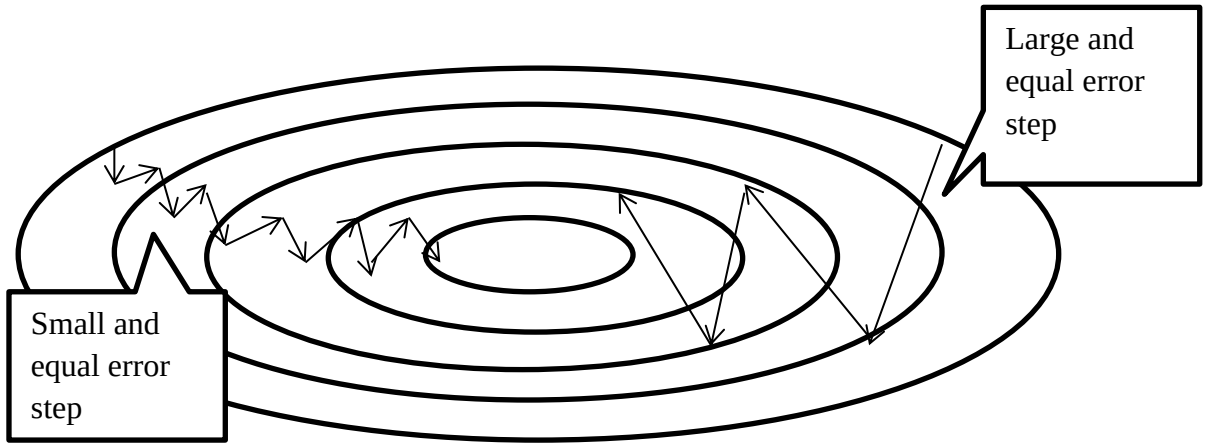

Figure 9– Error surface formed as an ellipse. The red trajectory is from an algorithm with large step size independent of surface shape, which leads to oscillation. The blue trajectory is from an algorithm with small step size, but equal magnitude in both directions. The small step size gives a good end result, but is very slow the large step size is much faster but gives inaccurate results.

The LMA method makes it possible to take small steps to avoid oscillating out of minima, but it is still not guaranteed to find the global optima. This is because the algorithm takes a step in the direction of the gradient of the error surface (see Figure 9). The problem is that the neural network cost function is not convex, which means it can have multiple local optima and it is impossible to know for sure that the optima the algorithm is converging towards also is the global optima, but this is never a guarantee with neural networks.

3.3.3 Support vector machine (SVM)

In order to use a SVM it is often not necessary to understand all the mathematical aspects behind it. This is also one of the reasons why SVM is sometimes used as “black box”, where the user does not really know how it works or what it has actually learned. A SVM can actually work very well without actually understanding anything about it, because of the robustness and its ability to solve a wide variety of tasks, without actually tuning the model for the problem. In order to keep this short only some key points behind SVMs will be explained without going too far into the mathematical reasoning.

SVM is a very popular algorithm used for a wide variety of problems including regression problems. There have been many iterations of improvement for SVMs from its early designs [34]. The first SVMs were only used for classification and all the model had to do was to find the optimal hyperplane that separate one set of data points from another. Hyperplane is a plain

in n dimension, or if the space is n dimensions the plain will have n-1 dimensions. The hyperplane is also called the decision surface, since based on which side a sample lies it will be one class or the other. The decision surface gives a linear separation of the space into two halves.

Where the decision surface is placed depends on the support vectors. The support vectors are the samples that lie closest to the decision surface. Using these support vectors the SVM will try to create a decision boundary that maximizes the distance/margin to the support vectors. The idea behind the support vectors is to find the decision boundary that separates the data best. When defining what is best we use the intuition that the decision boundary should lie as far as possible from the samples, because this means we maximize the margin before a samples is predicted with the wrong label/value.

The support vector gives us a good way to linearly separate two sets of data points. If however the data is not linearly separable we use something called a kernel function. The idea is to map data into higher dimensions, in such a way that it can be linearly separated, see Figure 10. Using enough dimensions it is possible to separate any set of points (because of infinite VC dimension)[35]. There are many ways to map the data and which method used, depends on the problem at hand.

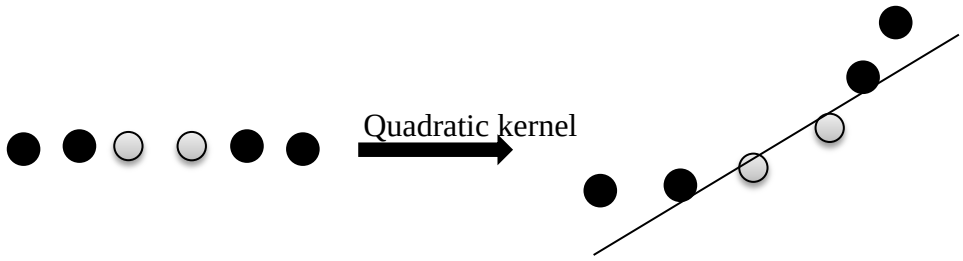

Figure 10 – A quadratic mapping makes the data linearly separable

SVM regression (SVR)

Support vector machines can also be used for regression problems, with some modifications. The idea behind SVR is very similar to SVM, the same math and kernel functions can be used. One of the main differences is that instead of having support vectors we now use an $\in$-intensive loss function. This means that we ignore errors less than $\in$, since $\in$ is normally set to a low value this mean that we ignore small errors. The point of this variable is to make it feasible to find the global optima of the SVR problem. An example of how the $\in$ influence the fitting can be seen in Figure 11.

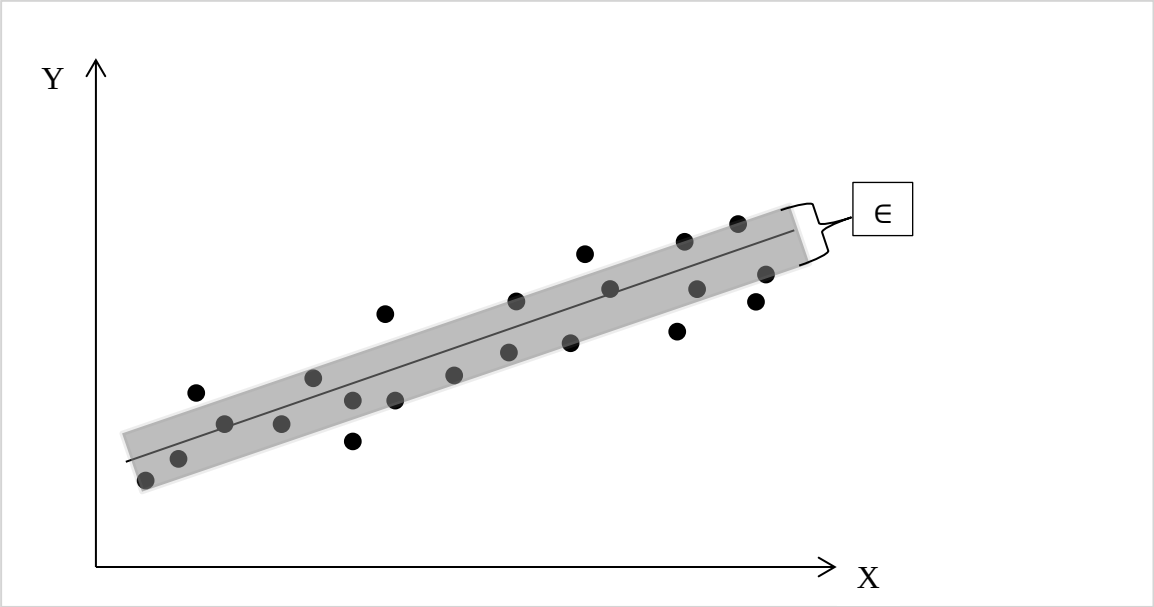

Figure 11 – Example of a SRV fitting some data. The grey area is ∈ and values inside will be ignored and the values outside are weighted proportional to the error.

SVM training

The math behind creating a SVM that can learn some function $h(x) = \langle w, x \rangle$, is to find some linear function that fit every sample as good as possible while also maximizing $\frac{2}{\|w\|}$. This gives a function that fit the data as well as possible while still be as simple as possible. w can be defined as following:

$$w(a) = \sum_i a_i - \frac{1}{2}\sum_i \sum_j a_i a_j y_i y_j K(x_i, x_j)$$

K is here some kernel like a quadratic function, but it can also simply be a linear kernel. The kernel function can be thought of as similarity function between two vectors. A detailed explanation for the role of a is outside the scope of this thesis, but it is Lagrange multipliers, which works as a scaling factor for the gradient under training

For SVM there is no need to choose a training function because an SVM will unlike ANN always find the global optima. However there is a chance that optimizing the functions is not feasible, meaning that it can sometimes be impossible to find the optimal line with the given constraints. This is why we introduce a slack variable or a way to accept some errors. How much error we accept is given by a parameter called C, this is related to ∈ for SVR, but it is not the same. A large C will accept a large error which may lead to a poor fitting; a too low C

will create a plane that fit the data poorly. The C value is related to the complexity of the model and is also important to reduce the effect of underfitting and overfitting (see 3.4.2).

With an SVM it is important to choose a good kernel, as described in 3.1.1 the kernel is used to map the data into some space where it is linearly separable. A common kernel is the radial basis function (RBF)[36]. The RBF gives a sort of Euclidian distance measure, with a parameter gamma that controls how fast the distance measure goes to zero as the distance increase. In practice this means that each sample has a sort of influence area controlled by gamma. Higher gamma gives a faster decay of the value and smaller influence; smaller value gives slower decay and a larger area of influence. A very high value will create a sort of average prediction over all values samples, while a very low value will create a prediction based on very few samples.

Although some of this math is a bit complicated, at the end what all this actually gives us is an instance based learner, similar the other learners like K nearest neighbour, because the kernel function can be seen as a similarity function. One of the main differences is that we have a smart way of storing all the training data with the support vectors and we have a nice way to measure non-linear similarity using a kernel.

3.3.4 Ensemble learning and boosting

Sometimes when creating a classification model or regression model the features or data available does not have a perfect relationship with what we want to predict. We may have some data that give an indication that a certain value is probable, but sometimes the data is also ‘wrong’. Using a large amount of such data can lead to poor performance and generalization. If it is impossible to attain better data, some sort of ensemble method can sometimes be used to create reasonable good model in such situations.

The principle with ensemble learning is to create multiple prediction models and combine them somehow to make a prediction together. If each of the models have learned some simple model/rule, combining them can give a more complex model, that is still robust since each model have only contributed a small amount to the end result. For this to work it is necessary that the models have learned different rules and are not making the same mistakes.

A common way to do ensemble learning is called boosting [37]. Even though some of the theoretical background for why boosting algorithms works is a bit complicated, using it in practice is often very simple. First step is to train a number of prediction models, making sure there is some variation between the models by either training them on different data sets or different inputs. Then, after the models are trained, they are combined into one model. A common way to combine them is some sort of voting or averaging [37], such that the value that most of the models believe is correct is chosen as the output. This means that if each

model are often quite accurate, but also fails sometimes we may get a model that is a bit less accurate in some cases but fails far less when using boosting.

3.4 Machine learning in practice

This section takes on some of the more practical aspects of machine learning. First, some information on what machine learning is used for today in medicine (3.4.1). Then, some practical challenging is discussed related to machine learning (3.4.2) and some ways to deal with these problems (3.4.3 and 3.4.4).

3.4.1 Healthcare informatics

In health care there is a large amount of data being stored, all from medical records, medications and data from medical image systems such as CT, ultrasound and MRI. With all this data available and the wish to decrease the cost of personalized healthcare there has become a need for applications that can manage all the data. The systems are not limited anymore to tasks like storing and retrieving patient data. There are also systems for diagnostic, like cancer detection and classification [3, 4]. This change has led to a machine learning paradigm in medicine, where algorithms use historical clinical data to find patterns or make prediction about new patients. If an algorithm is successfully created it may reduce the need of experts or assist experts at tasks that are especially difficult for humans.

Garg [6] reviewed a controlled trial, trying to assess the effects of computerized clinical decision support systems (CDSSs) and identify their benefits. CDSS are systems designed to aid clinical decision making. The staffs enter patient data into the system and the system makes a recommendation and prediction regarding the patient. Such systems normally use some sort of knowledge database containing other patients. Garg found that that many of the CDSSs improved the performance of practitioners in 62 out of 97 cases, but the effect on patient outcomes was unclear.

There has also been trials done in the area of cochlear implants. ECAP is as discussed in 2.4.1 a measure for the threshold for when the action potential is triggered from electrical stimuli, it is however not always easy to spot whether there actually are a response or if it is just noise. Charasse [5] investigated the possibility to utilize an artificial neural network to further automate the process of finding the ECAP threshold. They used a single layer network with 30 inputs corresponding to a 30 point ECAP trace; it had 5 output neurons corresponding to the various ECAP patterns. The network was trained with an adaptive backpropagation algorithm until an error of 10^{-6} was reached. They found that the neural network could identify the ECAP threshold just as good as an expert, but much faster. Similar trials have been done using decision trees and it is being successfully used today to identify ECAP [38].

3.4.2 Overfitting

Section 3.2 explained why we get the best hypothesis when minimizing the sum of squared error. It also explained a few faults in the assumptions that were used to find the most likely hypothesis. This section will explain one of the problems one might get when these assumptions do not hold.

Generalization

When trying to utilize some machine learning methods it is not enough to have the data, it is also necessary with a good way to create the model. One of the most challenging and important things to consider when creating a model, is to create a model that generalize well. Generalization means here how well the model created performs after training and it is tested on some new data. Getting good generalization is specially challenging on small datasets containing large amount of noise. Noise can lead the learning algorithm into believing it has found a pattern that it can exploit to predict the output, but the truth is that the pattern originated simply by chance. If the model have fitted noise or some pattern that only exist in the training data the model is likely to fail on new data, this is called poor generalization.

High bias vs high variance

When fitting a model it is often not clear how complex function is required for the model to fit the data well. Model complexity means here the highest complexity of a function the model is able approximate. This can be adjusted by a various number of ways, like the number of neurons in an ANN or the kernel in a SVM (see 3.3). Choosing the wrong sort of model can sometimes lead the algorithm to perform very poorly. This problem can sometimes be because of a high variance or high bias problem (or even both).

Sometimes when fitting a function to some data the final model may performs poorly simply because the model is too “simple” to fit the data, this is called underfitting or high bias. The opposite problem occurs when trying to fit a very model to a very simple data set, we then get something that does fit the data well, but it is an overly complicated function for the data, this is called overfitting or high variance. An example of underfitting and overfitting can be seen in Figure 12.

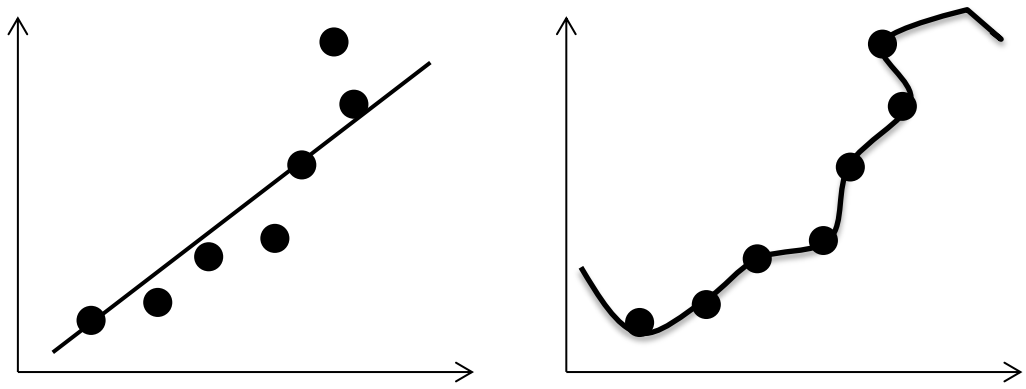

Figure 12 - Two possible ways to fit some data. The left function may be underfitting, while the one of the right seems to be overfitting. The function fit all the data perfectly and will give zero training error, but new data is unlikely to perform as good.

Underfitting or high bias normally happens when some sort of assumption about the data is too strong for the model to fit the data well. If we e.g. assume that the data should fit a linear model, but in truth it requires a much more complex function, the model will have a high bias towards a linear model and the model is not able to fit the data well. This does not mean that fitting a linear model to non-linear data is always a bad idea. Even if we know the data comes from some complex model or a high polynomial function it can still be worth to use a lower polynomial function or a linear model. This means that the model will underfit the data, but it may be worth doing so to avoid overfitting the data a lot.

Overfitting or high variance often happens when we use a too powerful/complex algorithm to fit some much simpler data. Intuitively it could seem that fitting the data as close to perfect as possible is the best way to go. However as explained in 3.2, the target value is usually affected be some noise. When fitting some data it is impossible to separate noise from actual information, which means that it is impossible to separate noisy data with low complexity from complex data with low noise. Fitting a complex model to data with high noise is very likely to give poor results, because the models is then likely to use noise when predicting unseen samples, an example of this can be seen in the right part of Figure 12.

It is not only noise that can cause overfitting, even if the data is completely noise free and a function that have the same degree as the true underlying function is used, it is still possible to overfit the data. This can happen in cases where the data has low variance or few samples are available, since there are multiple ways to fit a function to the same data. The less data samples that are available under training the more different functions the models is able to fit and the more likely the model is to fit the wrong function. This means that the complexity of the model should be based on both the quality of the data and the number of samples available in order to avoid overfitting [39].

3.4.3 Dealing with overfitting

The previous section showed some devastating effect of overfitting. Dealing with overfitting is therefore a crucial part of training a model. As mentioned having a large dataset can help to reduce overfitting, but if a large dataset is not available we need other ways to deal with overfitting. There are in practice two main ways to deal with overfitting – validation and regularization.

Regularization

When using regularization to avoid overfitting, the main focus is on the parameters the algorithm use to fit the function. Precisely how parameters are regularized depends on the algorithms, but in general when using regularization the goal is to reduce the magnitude of each parameter. This means that we make sure that none of the parameters contribute too much, because large parameters that contribute a lot to the end result is more likely to lead to overfitting[40]. Section 3.2 explained how minimizing the sum of squared error gives us the most likely hypothesis. This is only true if the prior probabilities for all the hypotheses are uniform. If this is not true we need to take each prior probability into consideration when finding the most likely hypothesis. In real life problems it turns out that 'small' hypothesis is more likely to be the correct hypothesis than larger ones. This implies that to find the most likely hypothesis we need to find the smallest hypothesis that fits the data "good enough". This is also stated by the popular principle of Occam's razor, which says *"Entities should not be multiplied unnecessarily."* Which is commonly interpreted as, the simplest hypothesis should be preferred over more complex ones [41].

What smaller hypothesis means depends on context and what sort of functions and algorithms we are using. It is therefore a variant of ways to make algorithms prefer the smaller and simpler functions. In general we want to encourage smaller hypothesis and not enforce it, in case we actually do need a more complex hypothesis. When we penalize the model in such a way, we actually 'handicap' the model, which actually tends to increase the bias and make the model underfit more. By using regularization it is important to find the balance between underfitting and overfitting. One example of regularization was shown in the neural network cost function in 3.1.1, where the weights are penalized proportional to their size. A similar idea was also shown in the cost function for SVMs (see 3.3.3)

Validation

When using validation to avoid overfitting the idea is to simply check whether the model is overfitting instead trying to prevent the algorithm from overfitting, as with regularization. Validation takes a small part of the dataset that was not used during training and uses it to estimate the model error on new samples (also called out of sample error). Depending on the size of the validation set this estimate can be a very accurate as long as this data set is not

used for any part of the model training. If the model performs poorly on the validation set it is also likely to perform poorly on new samples. Looking at the validation error can give some indications for how the model should be changed to perform better. If the model seems to be overfitting we could e.g. apply some regularization method. After we have used the validation set for testing the performance and made some changes, it is important to realise that the validation set is no longer "unseen" by the algorithm since we have now made a decision based on the validation data. If we want to repeat this process by iteratively make improvements, the estimated error may become overly positive.

Determining if the problem is overfitting or underfitting

Even though overfitting and underfitting are two opposite problems, it is not always easy to determine which is causing problems. If the problem is underfitting we probably want to increase the complexity of our model, but if the problem is overfitting we may want to reduce the complexity or perform some of the regularization methods mentioned above. What characterizes overfitting is that we have low training error and high test error, because the model have fit the training data very well but have been unable generalize. Underfitting however happens when the model is unable to fit the data because it is unable to learn a complex enough model; this gives both high training error and high test error. To actually figure out if the model is overfitting or underfitting and maybe even find how complex model is needed, it can be useful to plot the error as a function of model complexity, an example of this can be seen in Figure 13. Functions may not be this smooth in reality but it should give some indication for weather the model is overfitting or underfitting. As we can see in the figure the good fit is somewhere in between underfitting and overfitting, where the test error is as close as possible to the training error and the training error is as low as possible.

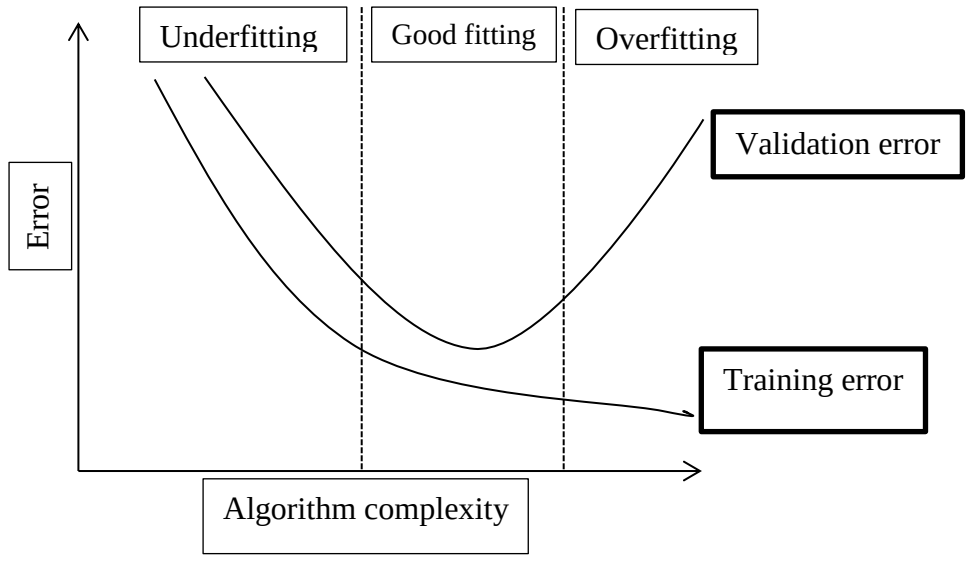

Figure 13 - A plot for how validation error and training error may change as model complexity increases.

3.4.4 Test data

Overfitting can happen in wide variety of situations; even overfitting of validation set is possible as discussed in the previous section. This is why we use a test set to confirm the performance of the model. If we had unlimited data, test data would not be of any concern, then we could simply train and test on the same data and the error we get would still be the true error for the entire distribution. The situations is very different for small datasets, since it is then a risk that the data available is only a small fraction of all possible samples, which makes a test set very important. If we do not have a test set and naively use the training error as the final score, it is likely to give an overly optimistic result. This is why it is very common to use a holdout method, where we "hold back" a part of the dataset under training in order to use it for testing on the final model.

It is important that the test data is representative for new samples in such a way that the algorithm has no information about it and has never seen the samples before. It is very important that this condition is met because the test set is what gives an indication for how successful the model will be on new data. We do not really care if a model is able to learn to predict the training data well. Getting good results on the training data does not say much, since it is not hard for a model to predict a value it has previously seen. In fact a success rate of 100% on the training set is likely to be an indication of overfitting. At least in real world problems there is always some noise or irregularities which make a perfect fitting very unlikely and the likely reason for the perfect fit is that the model is heavily overfitted.

To make sure that the test data resembles the scenario of getting new and previously unseen data it is important that the test data is totally unseen by the algorithm, this means that there is no decisions or parameters that is based on what is in the test set. Assuming that our dataset has a similar distribution to that of the distribution of all patients we can simply choose our test set as random samples from the dataset. The number of samples is a trade-off between accurate estimation of the test error and how well the model performs. Taking too much data for the test set will make the training data small and since the performance of a model is affected by the size of training data[42], this is likely to give an overly negative result. Making the test set too small will make the estimated error inaccurate and we may end up with an error that is much higher or lower than the true error.

4 Data analysis and pre-processing

This chapter will analyse the data available from the patient database. First, there is a description of the database available (4.1). Then, a general analysis will be performed, to improve understanding of the data and detect possible patterns (4.2). In the end of this chapter a few ways to pre-process the data is proposed (4.3) and a procedure for creating the prediction model is proposed (4.4), based on the results from the analysis.

4.1 The database

All the data mentioned in 2.4 is collected into a database, where it is possible to retrieve the measured ESRT, ECAP and impedance together with the corresponding C- and T-levels found by the audiologist. There are two sets of measures of impedance, C-level and T-level, one from the initial programming and one from the stable programming found after multiple sessions. Each of the 22 electrodes have its own ESRT, ECAP and Impedance value, so each patient has 22x3 values. We also have data about the age of the patient, when they got the implant and when the fitting was performed, so the total number of data values is close to 70 for each patient. The database available has about 300 patients all with similar or comparable CI and all with the data mentioned above. Some patients have been removed from the database because the programming used was too special or something different than standard procedure was used.

4.1.1 Standardizing the data

There are many variations of cochlear implants, to make sure that the data available is comparable only patients with comparable device have been used. The Device type used in our data set is the Cochlear CI24. There are few variations of this CI, the electrode design is either standard Contour electrode array or the Contour Advanced Soft Tip. The housing type is either 24R or Freedom and 512. These variations should not have any effect on the data or stimulation strategy used.

A set of C-levels and T-level for all electrodes are called a MAP. A patient can have multiple MAPs depending on preferences. Having multiple MAPs let the patient change C- and T-level depending on what works best in the current situation. Since there is no information about which MAP the patient has used the C- and T-levels used in this thesis is average of all the MAPs for each patient.

The pulse width is not part of our dataset; instead the pulse width is normalized to 25µs for every value. As explained in 2.4, we can do this without distorting the value since adjusting the pulse width lets the device give more current in total. The normalized value is the current

the CI would have given if it was able to give enough current without changing the pulse width.

4.1.2 Unavailable data that could be useful

There is some data that most probable is relevant data for setting the level of current that we do not have access to. The reason for deafness is one of the things that could possibly have a big influence on what levels of current to use; another could be the duration of deafness. Studies have shown that ganglion cell count decreases during the time after loss of hearing [43, 44]. However one of the biggest influences on ganglion cell count has shown to be the cause of hearing loss[44]. Because if this it seems reasonable that the duration of deafness and the cause of hearing loss would be useful when predicting the correct C- and T-levels [45]. However in our dataset there are no data about the patient like illness or cause of deafness.

Even if the duration of deafness was available this is not simple to measure. The duration of deafness can vary within one patient, which means patients have a different duration of deafness for different frequencies. Usually deafness starts from high frequencies to low frequencies, but most patients receive an implant when low frequencies are lost. Meaning they can have been deaf on some frequency areas for many years, but only a short duration of deafness on other frequencies when they get the implant. The problem here is that patients that have not heard certain frequencies for many years might have problems getting used to these frequencies again, especially since the hearing now works with electrical stimulation using a CI. This makes it difficult predicting these levels, but this may not be such a big problem for the low frequencies.

While the ear is important for perceiving sound, the brain is also important for handling this information. How well we hear greatly depends on how well our brain is able to interpret the signals sent from the ear. It is possible to measure auditory cortical responses, which is a way to measure how the brain responds to sound. This measure gives an indication of how well the sound is perceived, but these kinds of measures have not been performed on the patients in the database available.

The problem is not only that data is completely lacking; the data we do have is sparse, in the sense that the set of objective measures do not have data for every electrode. This is mainly the case for the ESRT and ECAP values. To save time it is common practice to only measure every second electrode, this is however not consistent since sometimes the audiologist is not able the get measurement from an electrode and the next electrode is measured instead. The reason why the audiologist did not measure the electrode is not available, but it would be useful to know if the electrode was omitted at random or if there was a problem when trying the get the measure.

The limitations to our dataset are not only that there are missing data about physical attributes related to the patient. A major factor are also things like how cooperative the patient was during the programming sessions, which may influence if the C- and T-levels ends up with an optimal value. It is also possible that personal preferences influence C- and T-levels since they are mainly dependent on responses from the patient.

Factors that influence the results, but are not among the observed data can make it hard for a model to predict accurately. It is also possible that some of the non-observed factors actually are partly measured by the objective data.

4.2 Data exploration and analysis

As mentioned earlier the way data is coded and represented can have a crucial impact on how well the algorithm performs. Making non-linear transformation of the data can sometimes make the data more useful or easier to utilize by the prediction model. Since such transformation that algorithms do is often quite general and not tailored for the specific problem, it can be a good idea to do this manually if there are known ways to do this for the given problem. If one for example knows that a certain ratio between two variables is crucial to the outcome this ratio should be an input and not the two variables independently. For the data available there is no known way to combine or transform data in order to make the data easier to utilize in a prediction model. One possible method to find possible ways to transform or combine the data is to visually represent the data and see if it is possible to spot possible patterns. The next section will use statistical methods to analysis the data combined with some visual representation.

The data analysed in the next sections is the training data with objective measures and the C-/T-levels. Section 2.2.3 explained how the CI is programmed and some special problems related to programming the CI of infants and young children. These special problems can lead to some inaccuracy during the programming, this is why patients younger than 5 years will not be included in the training data or analysed here. When the patients mentioned are excluded from the data set we end up with 158 patients, which will be used for data analysis in the next section and for model training in Chapter 0. Note that a separate test set is already excluded from these 158 patients, which will be used at the end of Chapter 0 to verify the performance of the models.

4.2.1 Why understanding the data is important

One important note is that during the process of fitting a model we assume that we can deduce a general rule by looking at examples of input values with their corresponding output. This is not possible in the general case! Even if we have found a rule that fits all the samples perfectly we cannot say that our hypothesis is the correct hypothesis. The famous philosopher

David Hume was among the first to realize the problem of induction. His point was that it is for the general case impossible to generalize past the samples we have seen. A similar observation was later made by David Wolpert in his famous "No free lunch" theorem[46]. He claims in his theorem that no learner can perform better than random guessing in the general case. The reason for this is simply because for every model that fits the training data well we can always construct a test set that the model will fail on and we can construct a test set where the algorithm will get everything correct. If a problem is uniformly sampled from all possible problems both are equally likely and since we do not know anything about the test set it is impossible to know if we have the first or the latter. If we for example have the data 1, 2, 3 and 4 with the corresponding output 2 4 6 and 8. We can easily induce that the hypothesis function could be h(x) =2*x and this seems likely. But it could be that this only holds true for numbers less than 5 and past 5 there are something completely different. Since we only had 4 numbers it is not unlikely that our hypothesis could be wrong, but if we had thousand numbers or maybe a million numbers and our hypothesis still holds we feel more confident that our hypothesis actually is the correct one. The problem is that even if we have a million numbers there is probably a complex function that can get all the numbers we have seen exactly correct but still is not the correct hypothesis and it could in fact get every new sample wrong. Even though this is a very theoretical view similar conclusions have been drawn from empirical studies [47, 48]. These studies show that algorithm which is best, greatly depends on the problem and various parameters and there is no algorithm that performs best on every problem.

The point is that a machine learning model needs some domain knowledge. One can often assume that a problem from the real world is not drawn uniformly from all possible problems, which justify assumption like similar input lead to similar output, but the more domain knowledge we can put into the algorithm the better. To obtain domain knowledge it can be useful to analyse the available data and look for possible patterns to use and understand the data before giving it to some model learner. Understanding the data may also make it possible to give domain knowledge to the algorithm trough parameters, like the kernel function in a SVM, the number of neurons in a neural network as discussed in 3.1.1, or some sort of non-linear transformation of the data.

4.2.2 Variance analysis

The data available is on a scale from 0 to 255 for C- and T-levels. Investigating the variance will show how the values are spread out on the scale. Analyzing how values are spread out on the scale can help understand the data and possible detect faults in the data or detect outliers.

C- and T-level variance analysis

By looking at Figure 16 we can see that the variance is a bit larger for C-level than for T-levels, which may indicate that T-levels will be easier to predict. Comparing Figure 14 and

Figure 15 we can see T-levels is about 80CL lower than C-levels, which is excepted since C-levels are the upper limit and T-levels are the lower limit, as explained in 2.3.1.

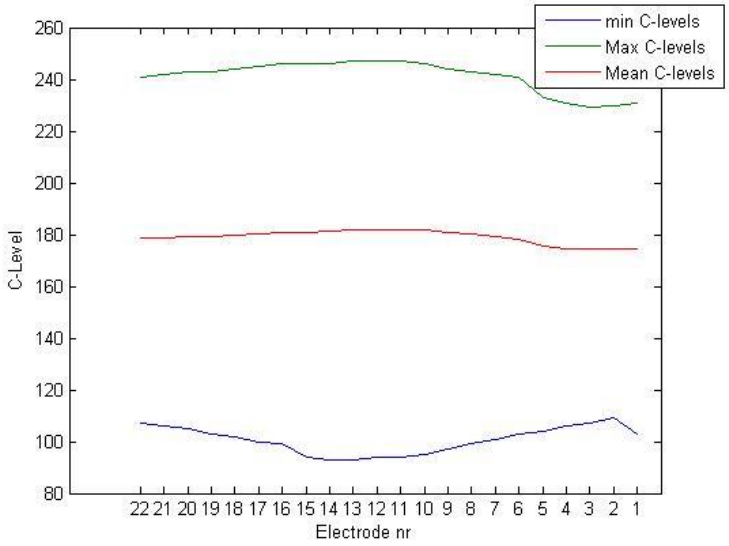

Figure 14 – Shows max, min and mean C-level for all the electrodes for all 158 patients

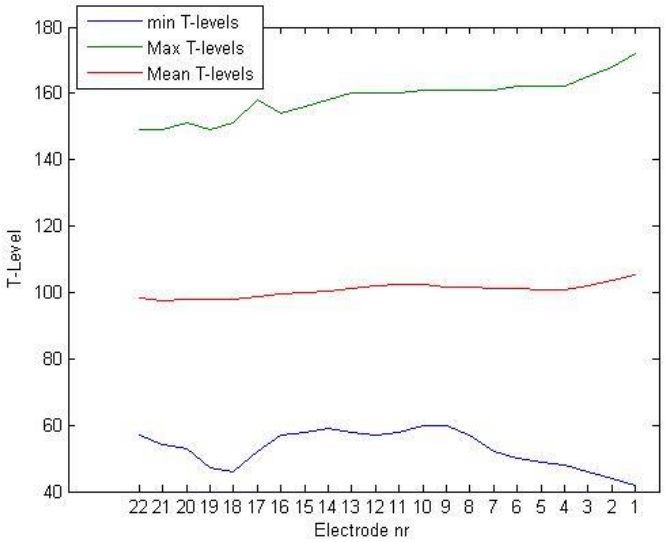

Figure 15- Shows max, min and mean T-level for all the electrodes for all 158 patients

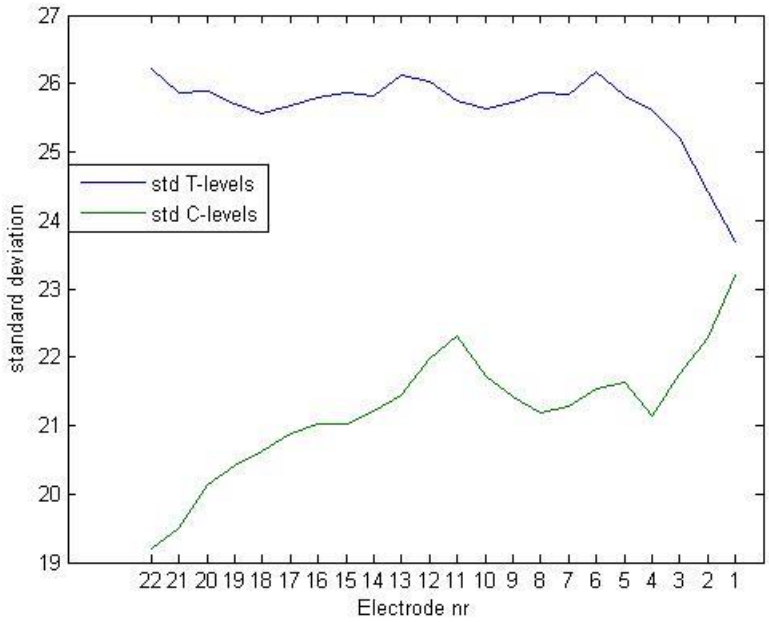

Figure 16– Standard deviation for C- and T-levels over all electrodes for all 158 patients. The plot shows the standard deviation, even though the values are on a scale from 0-255 the standard deviation is fairly small for both C- and T-levels. The standard deviation is about 25 for C-levels and about 21 for T-levels, this means that about 68% of the patients have a value that is -+ 25CL and -+ 21CL away from the mean for C- and T-levels, respectively.

A boxplot for C- and T-levels can be seen Figure 17 and Figure 18. The whiskers are defined in such a way that samples will get red crosses if they are outside the 99.3 percentile of the normal distribution. If we assume normal distribution, the samples outside the 99.3 percentile are very likely to be an outlier, especially if there no other samples nearby. This means that the sample could be different for some reasons and should not be treated like any other. Possible reasons could be noise, some sort of error with the data or simply an anomaly. Patients with these extreme values should be investigated further and maybe discarded from the dataset, if they actually turn out to be outliers. The box plot shows that most of the values seem to be inside the 99.3 percentile, but both C- and T-levels have a few samples that seem to be a bit lower than most of the others. These potential outliers will be discussed further in the pre-processing step (4.3).

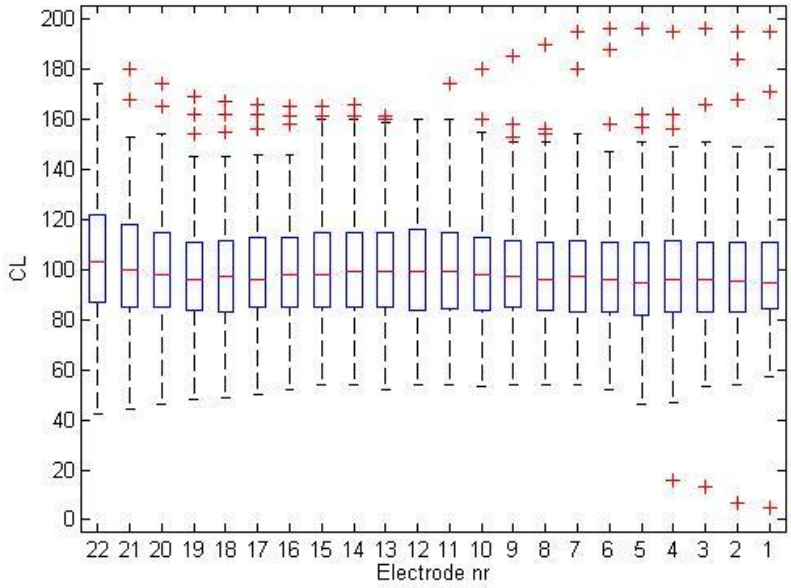

Figure 17 – boxplot for T-levels over electrodes 22 – 1 for all 158 patients. The crosses are values outside of the whisker. The whiskers cover values that lie within the following range: smaller than q3 + 1.5(q3 – q1) or greater than q1 – 1.5(q3 – q1), where q3 and q1 are the 25th and 75th percentiles, respectively.

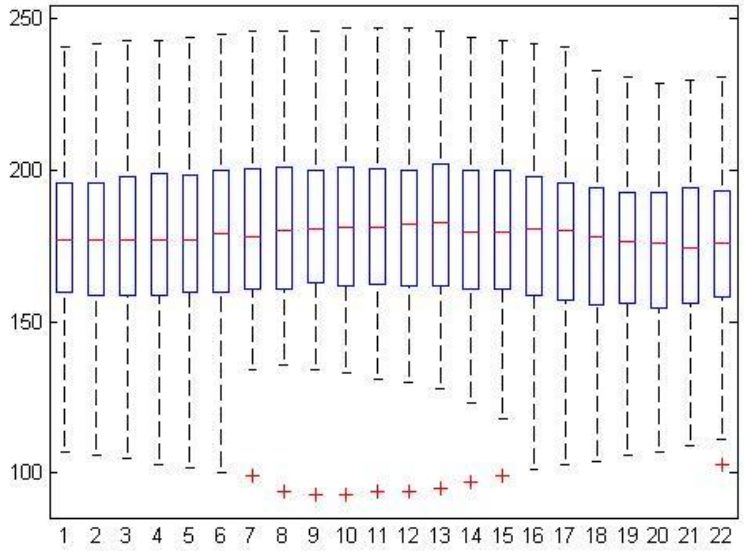

Figure 18 – Boxplot for C-levels of electrode 22 – 1 all 158 patients. The crosses are values outside of the whisker. The whiskers cover values that lie within the following range: smaller than q3 + 1.5(q3 – q1) or greater than q1 – 1.5(q3 – q1), where q3 and q1 are the 25th and 75th percentiles, respectively.

Impedance variance analysis

Impedance is as explained in 2.4 among other things used to determine if an electrode is operating properly. One way to tell whether an electrode is not working as it should is if the impedance of that electrode is very high. It is common to turn of electrodes with 15000 – 20000 ohms impedance and the software used to program the CI automatically turns off electrodes with impedance higher than 20000 ohms. Figure 19 shows a boxplot for impedance, there seem to be some extreme values that are far away from most of the other values. All these values have impedances higher than 20000 ohms and their C- and T-level are not set. It is safe to assume that these values are outliers and to avoid that they cause problems for other analysis and when training the model, these values will be removed from the dataset.

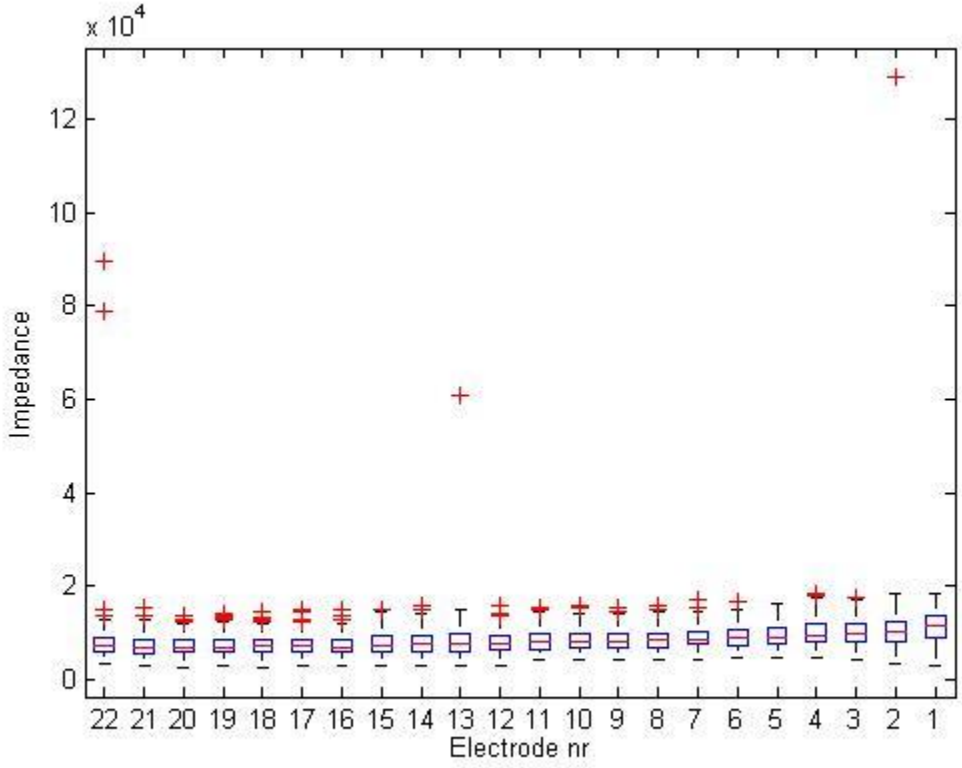

Figure 19 – Box plot for impedance over all the electrodes 22 -1 for all 158 patients. The whiskers cover values that lie within the following range: smaller than q3 + 1.5(q3 – q1) or greater than q1 – 1.5(q3 – q1), where q3 and q1 are the 25th and 75th percentiles, respectively.

4.2.3 Data covariance analysis

When performing measurements in a system, there is a chance that some of the measurements actually measure the same component. If we for example have one measure of something in feet and another in meters, removing one of them would not remove any information. Having a 100% overlap is very easy to handle, it is however harder when the data contain some

overlapping information. If we have the measure of speed on a car and the rpm of the wheels there is probably a large correlation, but it is probably not a perfect correlation.

To analyse the correlation of all the data it can be useful to use correlation matrices. A correlation matrix is a matrix with every pair of correlation coefficients. A correlation matrix R can be defined as following:

$$R(i,j) = \frac{C(i,j)}{\sqrt{C(i,j)C(j,j)}}$$

$$C(x_1, x_2) = E[(x_1 - \mu_1) * (x_2 - \mu_2)]$$

The covariance matrix with n data vectors will then give a $n * n$ large covariance matrix. $R(i,j)$ will be the correlation between data vector i and data vector j. Visualizing the correlation coefficients in a colour map helps to localize interesting values, correlation maps for ECAP and ESRT is shown in Figure 20 and Figure 21, respectively. These correlation plots only contain coefficients for even numbered electrodes and electrode 1. This is because there are a very large number of missing values for all the other electrodes (as explained 4.1.2). Finding correlation when there are very few values can give inaccurate results.

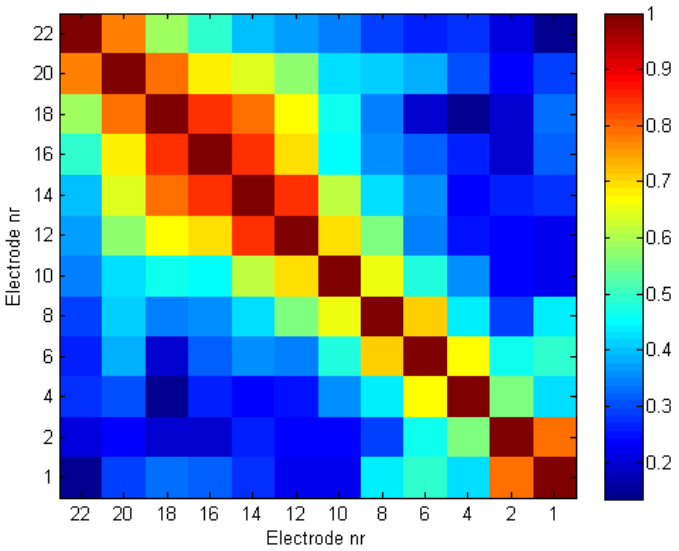

Figure 20- Correlation plot for ECAP

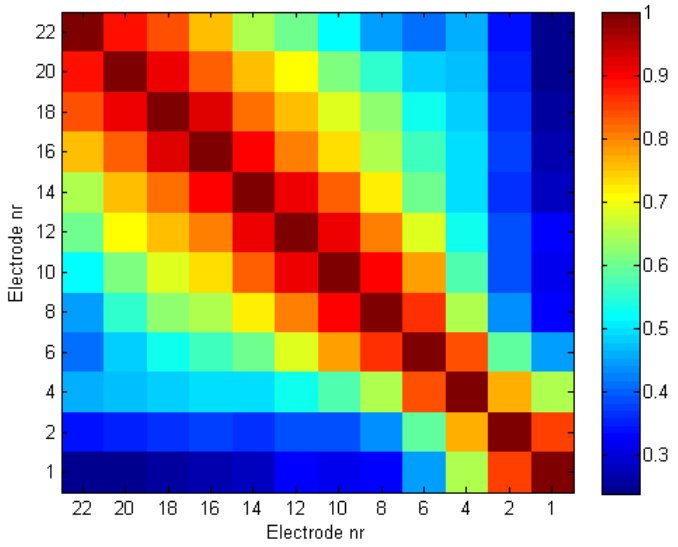

Figure 21 - Correlation plot for ESRT

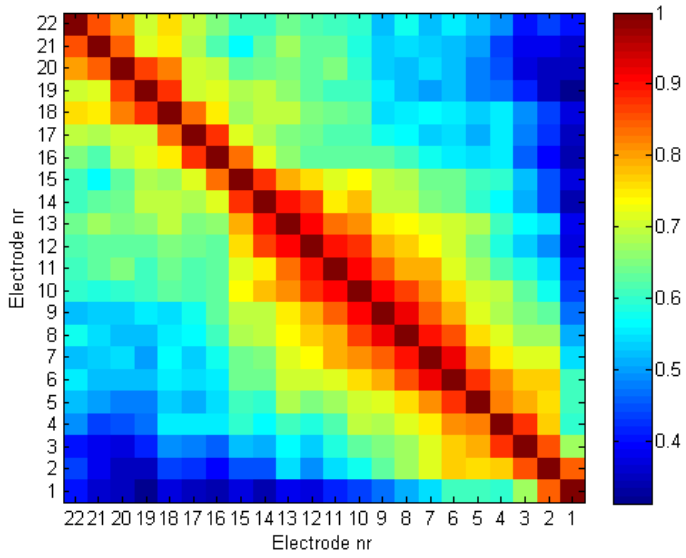

Figure 22 - Correlation plot for impedance

From these plots it is easy to spot a few obvious patterns. All maps show high values around the diagonal, this means that electrodes close to each other have high correlation. In the upper left and the lower right corner are the correlation values for the electrodes furthest apart, they show very low correlation. The overlap in information may actually turn out to be a good thing, since neighbouring electrodes show high correlated values we may be able to restore missing values by using this fact. Using this to restore missing values will be discussed further in 4.3.1.

4.2.4 C- and T-level correlation with data

By analysing the correlation between the objective data and T- and C-level we may get some indication for what we can expect from the various objective measures. The correlation analysis will use Pearson correlation, which means that only linear dependencies will be accounted for. The correlation coefficients will not be a good estimate for models with non-linear function approximation, but may still give some indication for what we can expect.

C-level correlations

From Figure 23 we can see that the objective measures have some correlation with C-levels, but they are not good enough to make a good prediction from only one of the objective measures. The correlation with age is very poor which may indicate that a linear model will not have much use for it. The correlation coefficient also depends on the electrode number, meaning that some electrodes have better correlation between objective measures and C-level than others. The result of this may be that C-levels for some electrodes are easier to predict than for others.

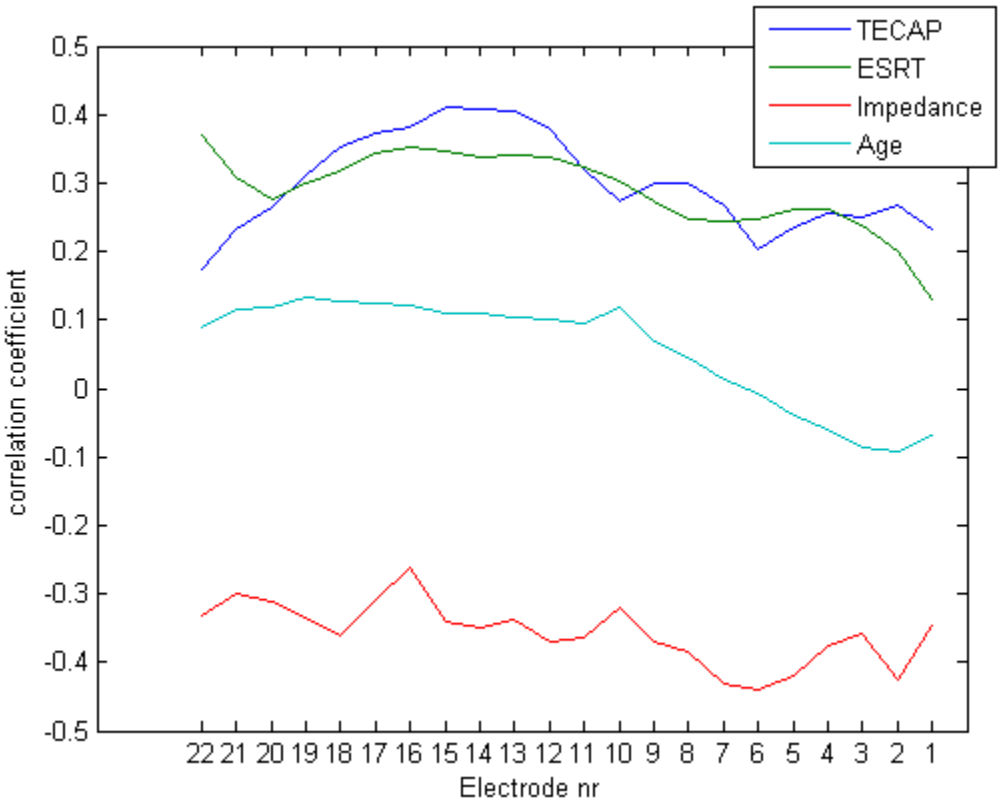

Figure 23 - Correlations between the objective measures and C-levels.

T-level Correlation

A similar plot as for C-levels is shown for T-levels in Figure 24. The correlations are generally lower for T-levels compared to C-levels; this is especially noticeable for impedance. For electrode 22 to 8 the correlation is very poor, but for electrodes 7 to 1 it is a bit better. Age has also very poor correlation with T-levels, although some electrodes seem to be a bit better than others. The variance analysis (4.2.2), showed that the average variance for T-levels were a bit lower than for C-levels, which could imply that T-levels will be easier to predict than C-levels. However, from this correlation analysis it seems that the objective data is more correlated with C-level than with T-levels which could make it easier to predict C-levels.

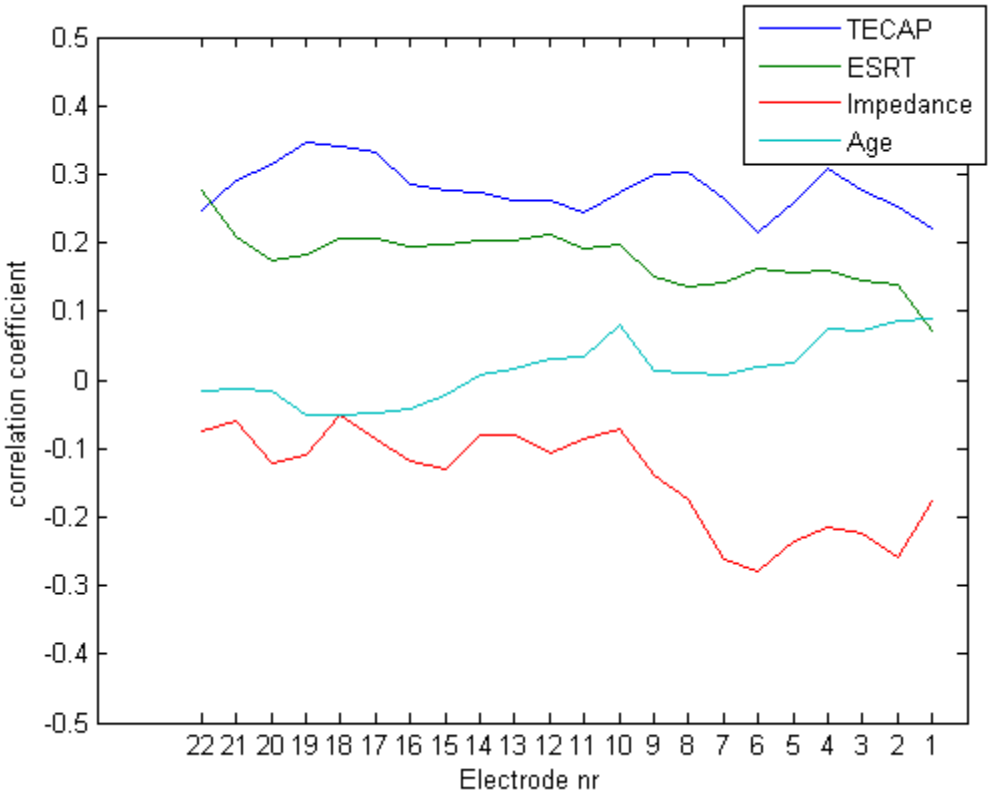

Figure 24 – Correlation between the objective measures and T-levels

4.3 Pre-processing

When dealing with data related to humans it is common that the data can be noisy or have a large spread/variance. This can makes it very challenging for a model and this is why it is important to make sure the data is at an optimal state before we start training a model. This

does not only mean that the data should be relevant and as noise free as possible, but it is also important that the data is on a good format and scale. The following section will discuss ways to prepare the data for training based on insight from section 4.2.2.

4.3.1 Restoring missing values

Since the dataset have many missing attributes, it has to be managed somehow. Some sort of imputation is a common way to handle it. Imputations methods are commonly used on data in medicine, since missing data for various reasons are quite common. Data can sometimes be hard or costly to obtain, especially in medicine. So being able to handle datasets that have some missing values can be necessary. Being able to handle samples with missing data let us keep the dataset as large as possible. There is also the problem when dealing with real patients one cannot expect everything to be perfect and a few missing values should not be enough to discard a patient. Since missing values are quite common it has been created a multiple ways to deal with the problem, Sterne, J.A.C., et al [49] discuss various statistical methods to handle missing data in epidemical and clinical research. These methods could also be used on the dataset available here. While some methods focus on restoring the variables the focus here is to make the model predict as accurate as possible and not to make valid interference, since we have no need for the correct data in itself, all we want to achieve is for the model to predict as well as possible.

As discussed in 4.1.2 the norm in the database is that about half the electrodes have measures for ECAP and ESRT, but some have fewer and some have more. Some imputation methods simply set the missing values to the average over all values of that attribute, but from the correlations in 4.2.3 it seem that it should be possible to do better. By looking at the correlation matrix in Figure 20, Figure 21 and Figure 22, it is clear that close by electrodes have fairly similar values. This means that if we have the adjacent value of missing value we should be able to find a reasonable accurate estimate for the missing one. One way to use neighbouring values to restore missing values is to use some sort of interpolation. Interpolation is a method that uses known values to create some function that fit the known data; we can then use this function to estimate unknown values between the know values.

The number of missing values varies from sample to sample, some have a measure on almost every electrode and some have on almost none. Because of this we may end up interpolating with very few values if there is no requirement for how many missing values we allow. We want to avoid interpolating with very few values since it is likely to yield poor results. Many missing values may also be an indication of possible outlier. Even though we do not know the reason why values are missing it is reasonable to assume that many missing values could have been caused by some complication or anomaly. To avoid interpolating these samples it needs to be a limit to how many missing values that should be allowed. Having at least 8 samples of each type of measure seems reasonable; this means that the allowed number of missing values

for each measure is 14 for each objective measure; there are 20 patients that do not fulfil this requirement.

Most samples in the dataset have objective measures for electrode 1 and 22 (88.6% and 88.1% for electrode 1 and 22, respectively). But this also means that there is a fair amount that does not have a value and without extrapolation these would have to be discarded. While extrapolation generally is harder than interpolating values, this seems necessary to avoid excluding too many samples.

The interpolation is performed with MATLAB using the function interp1 with Piecewise Cubic Hermite Interpolating Polynomial (PCHIP). PCHIP interpolates using the following criteria[4]:

On each subinterval $x_k \leq x \leq x_{k+1}$ *,* $P(x)$ *is the cubic Hermite interpolant to the given values and certain slopes at the two endpoints.*

$P(x)$ *interpolates y, i.e.,* $P(x_j) = y_j$ *, and the first derivative* $P'(x_j)$ *is continuous.* $P'(x_j)$ *is probably not continuous; there may be jumps at the* x_j*.*

The slopes at the x_j *are chosen in such a way that* $P(x)$ *preserves the shape of the data and respects monotonicity. This means that, on intervals where the data are monotonic, so is* $P(x)$*; at points where the data has a local extremum, so does* $P(x)$

4.3.2 Removing samples

I will here discuss and explain patients that have been removed from the data set. It is important to be very selective and careful when removing samples unless it is clear why the samples should be removed. Removing samples that are not outliers may distort the distribution or removing samples could discard data that is important for the model to find a good hypothesis.

Section 2.3.1 discussed issues related to finding the C- and T-levels for young children and infants, because of the lack of feedback. This means that the C- and T-levels can be a bit off from what they really should be for young children. Even for adults that are capable of giving good feedback it can take years to find an optimal CI programming. When training a model we want to avoid having training data with suboptimal results, because then the model will learn to find suboptimal results. This is why patients younger than 5 years are excluded from the training data. Older patients often get a good programming after about a year, so patients that have had the CI for less than a year may also have a suboptimal fitting. Therefore patients that have had the CI for less than a year will also be excluded from the dataset.

[4] http://www.mathworks.se/help/matlab/ref/pchip.html

As discussed in 4.2.2 there seems to be some patients that lie outside the 99.3 percentile for C-levels, this could be an indication of outliers. Applying the pre-processing method described in 4.3.1, with the criteria that samples with more than 14 missing values among one of the objective measures would be discarded, actually removes a few of the outliers. After pre-processing we get the boxplot in Figure 25. The boxplot shows that there are still some possible outliers and all of these outliers belong to two patients. The one outlier (red cross) on electrode 22 belongs to one patient, the rest belongs to another. Even though these values seem a bit low there are no obvious faults or abnormalities in their objective measures. It thus seems that there is not sufficient reason to remove the samples.

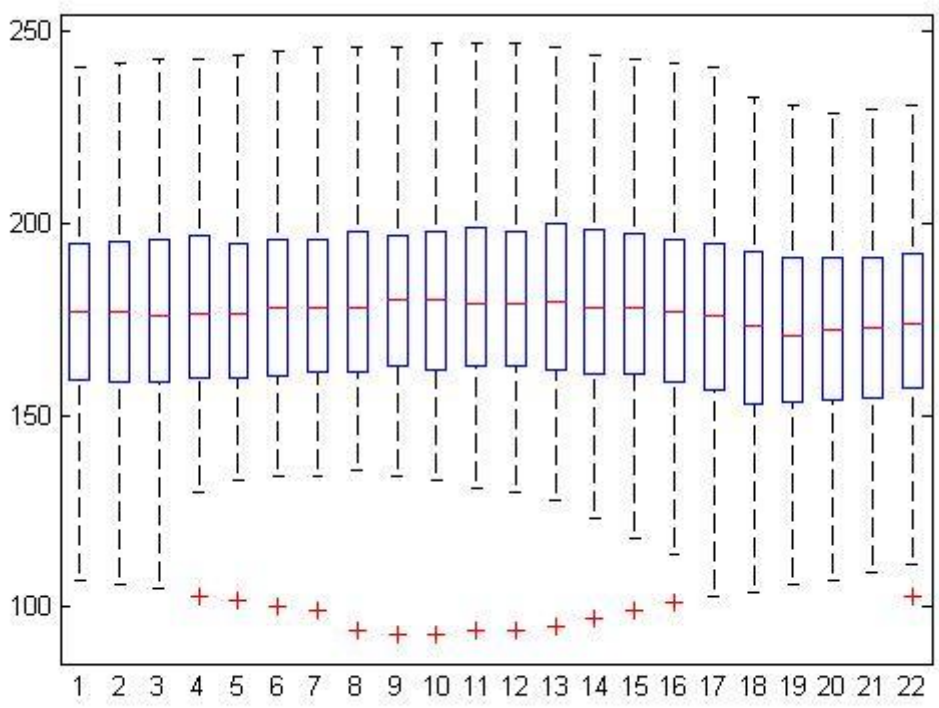

Figure 25 – Boxplot of C-levels after pre-processing

4.3.3 Feature extraction

When talking about the input data to a prediction model it is common practice to refer to it as predictor or features. The features used to learn a model is a crucial part of creating a successful prediction model. The following section will discuss a number of methods that can be used to create features from the available data.

Why we do feature engineering

Feature extraction or feature engineering as it is also called is one of the most important parts to successfully create a model with low prediction error. It may without further thought seem that just naively using all the features is the best option because then we use all the information available. The problem is that the more features we have the more computer power the model requires to train. This may not be such a big problem with our dataset, but with larger datasets it can be. A far larger problem is that in practice, using more features than what is necessary can increase the error rate of the model (as discussed in 3.4.2). Some models like a decision tree is robust to redundant features, since it only chooses features that separate the dataset when splitting. For similarity based models like K nearest neighbours, redundant features can pose a huge problem. Even when using a few extremely good features adding irrelevant features can make the performance very poor. The reason for this is that in high dimensional space most points are equally close to each other, this means that adding a new dimension that contains some noise can wrongfully move points further apart, which will lead to misclassification. Many irrelevant features can also lead to overfitting because the more features you have the greater chance there is that some of the features seems to separate the data well, but is actually nothing but noise, (see Figure 12). Adding features that contain random data will actually reduce training error, but as explained in 3.4.2, this will give very poor performance on new data.

Why feature engineering is hard

Applying machine learning to problems like natural language processing and image classification is something that people have been doing for a long time. This is also why there are so many ways to extract features in these fields. The methods are often created especially for that field which makes them not very applicable to other areas. This means that since it has been done very little work in the area of using ML for predicting C- and T-levels it does not exist any good features that has been created for this. Creating features is not a simple and forward task. Among the most famous features in image processing is the Scale-invariant feature transform (SIFT) feature created by David Lowe, it took him 10 years to create this feature and is so complex that very few people understand it. It is probably possible to find good features in less time than that, but crafting good features can be very hard. There has been in later years an increasing need for ways to extract features automatically because of the increasing popularity of big data. Since machine learning is now being applied to so many areas there is no time to handcraft features for them all. One method that tries to solve this is called deep learning[50]. The idea behind deep learning is to create a model that creates features automatically, simply by “looking” at large amounts of data. This method have proven very successful at a wide variety of problems, unfortunately it requires a large amount of data for it to be applicable, which makes it hard to apply to the dataset available here.

Principal Component Analysis (PCA)

There exist a few different methods that try to remove redundant components while keeping as much information as possible, one commonly used method is called principle component analysis (PCA). The idea behind PCA is to find some combination of the original components that gives a new component which contains as much variance as possible. We then repeat this, but making sure the new variables does not correlated with the previous ones. We keep doing this until enough of the data have been represented. This will often end up with a few variables that contain most of the information from the original data. By doing this we hope to achieve two things:

1) Remove the redundant data which gives us fewer features to work with

2) Assuming that noise represents less of the variance than the information we want, the resulting data may contain less noise.

One of the strengths of PCA is that it is non-parametric, which makes the result independent of the user and the result is purely data-driven. On the other side we may sometimes want to incorporate a-priori probabilities into the calculations. Another nice feature with PCA is that it always gives components that contain a lot of information. The problem with this is that this information can be completely useless at what we want to use it for. The only thing PCA takes into consideration is variance which means that irrelevant features will not necessarily be removed and relevant features will not necessarily be kept. This can happen when irrelevant information is the cause of the biggest variance in features space. It is also important to keep in mind that noise can have high variance, which means that if the components have very high noise it can actually remove the useful data and only retain the noise, but if there data has this much noise it probably was not very useful in the first place.

Feature 1, correcting for bias

When visually plotting the data it does not seem to be a clear relationship between the objective measures and C/T-levels. One possible reason could be that some unknown factor has influenced the values. If we assume that the factor have influenced all the electrodes equally and it is constant we remove this effect by simply centring the values cross electrodes. This is similar to the idea behind some of the predictors used in the model that won the Netflix grand prize [51]. The idea is to correct for some systematic preference by the user or some other factor that has influenced all the electrodes. This can be formulated as follows:

$$o'_{p,e} = o_{p,e} - \frac{\sum_{i=1}^{22} o_{p,i}}{22}$$

Where $o_{p,e}$ is objective measure of electode e from patient p. This means that we for all the

objective measures subtract the mean over the electrodes for each patient. This method will be tested alongside the unmodified objective data in 5.3.

Feature 2, noise reduction with PCA

Given the poor correlation between the objective data and C-\T-levels and that it does not seem to be any obvious patterns when plotting the objective that, it may contain a large amount of noise. One of the benefits as mentioned with PCA is the possibility to remove noise and redundant information. Section 4.2.3 showed that adjacent electrodes have a large correlation on all the electrodes (see Figure 20, Figure 21 and Figure 22). This implies that objective measures from adjacent electrodes measure some of the same component. This means that two objective measures from adjacent electrodes contain overlapping information. In a way it seems likely that objective measures from the electrode we want to predict T- and C-levels is the most relevant. On the other hand, since they are so heavily correlated they may actually measure almost the same component in addition to some noise. If this noise is not too large we could use PCA to create a new feature. This feature would be based on the objective measure from the electrode we want to predict T- and C-levels for in addition to objective measures from their adjacent electrodes; this process is demonstrated in Figure 26.

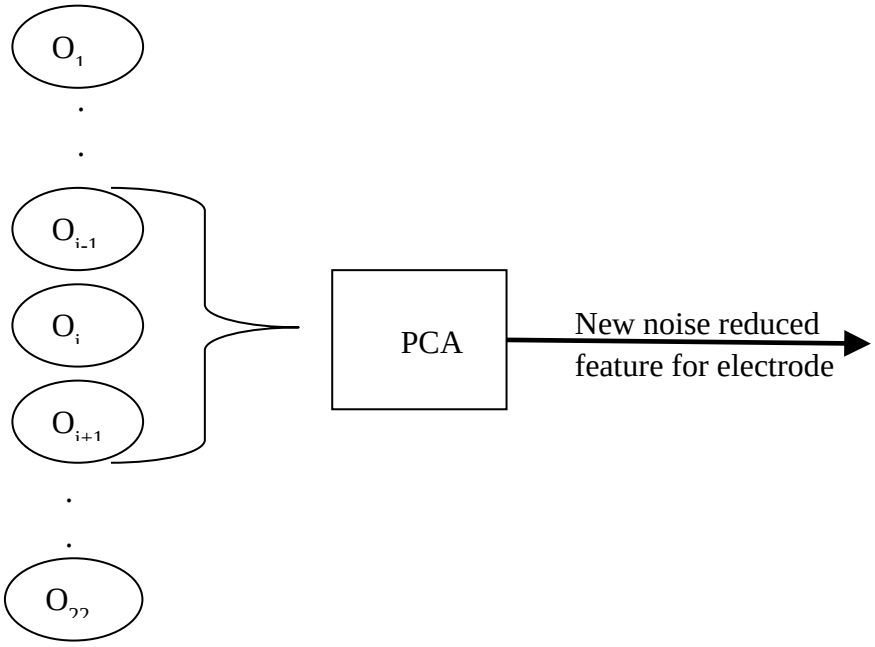

Figure 26 - Process for reducing noise for objective measures, where O_i is any objective measure for electrode i

This method will be applied to the objective data and tested alongside the unmodified objective data in 5.3.

4.3.4 Scaling features

After the data have been restored it can be useful to centre and scale the data. Scaling and centring the data makes sure that the various kinds of data are on a similar scale. This can be beneficial both to speed up the algorithm, but also to get better results. If the data is centred problems like elliptical error surface as in Figure 9 get much less likely. This is true for all algorithms that use some sort of gradient to decrease the error. This is also important for similarity measure based algorithms, because they can fail badly if the data is on very different scales. If one feature is on a much larger scale than any of the other features a change in it will count for most of the distance, while no matter what the values are for the other features it will not make any noticeable difference to the end result. This means that only one of the features actually is used by the algorithm.

While it is useful for the algorithms that the data is on the same scale, it is also useful when manually analysing the values and comparing the various input data. Rescaling the data makes it easier to plot the data together. Centring the data remove trends which makes it easier to spot if values are e.g. higher or lower than normal.

Before the data is used it will be centred and scaled as follows:

Centring

$$x' = x - mean(x)$$

Scaling

$$x' = \frac{x}{std(x)}$$

This method is used on all the data before the model train on it in chapter 0.

4.4 Procedure for creating the model

Until now various machine learning methods and the data available have been discusses separately. This section will discuss how machine learning methods and the patient data can be used together to make a model that can predict C- and T-levels.

4.4.1 Technical problem formulation

The problem we want to solve is as follows, we have 22 electrodes which all have two values that we want to predict (C- and T-level), meaning a total of 44 values. These levels are stable/final values that are found after about a year. Our input data that we can use to find these 44 values is the objective measure discussed earlier (ESRT, ECAP, impedance and age), which will be referred to as features. We have ESRT, ECAP and impedance data corresponding to each of the electrodes, but as explained in 4.1.2, ESRT and ECAP have been measured only on about half the electrodes. These measures are done on about every second electrode so using the values we know and interpolate them using the method explained in 4.3.1 should give fairly good estimates for the missing values, since neighbouring values are known to be fairly similar (see Figure 20 and Figure 21). This leaves us with 22 values for each objective measure meaning 66 plus age, which make a total of 67 input values.

Now, we know what the input and output should be, but we still need to figure out how to apply this data to a model. With a Neural Network we could use all 67 input neurons and 22 output neurons, but we do want to try a variety of models and most models can only give one output value. One possibility is to predict a single electrode and use the electrode number as input and let the algorithm figure out how to deal with that number. This could help if it is information that can be exploited between the electrodes. This means that information can be learned from one electrode and then be useful to predict other electrodes. The problem is that this would require a more complex model and since our training set is fairly small, it is smart to avoid making the model more complex. Another problem with this approach is that adjacent electrodes have very similar values meaning we would add multiple samples that are very similar; this would make the sample space look very strange since it would look like most samples are almost equal.

Another possible problem formulation is to look at the problem as 22 separate C-level problems and 22 separate T-level problems, with 67 inputs and 1 output. This means that predicting one value for each electrode is one problem. Using all 67 inputs seems a bit redundant for predicting one value, since only a small fraction of that number is data that is actually measured from the electrode we want to predict. Since measures from neighbouring electrodes are highly correlated it is likely that removing them will not remove a whole lot of useful information. If we only keep the data for the electrode we want to predict we end up with four features/inputs, corresponding to one output value (see Figure 27). The overview in Figure 27 shows only what features/data we have available to create the model, but it not certain that we using all the data is so we need a method for choosing what features to use, this we be discussed the two next sections.

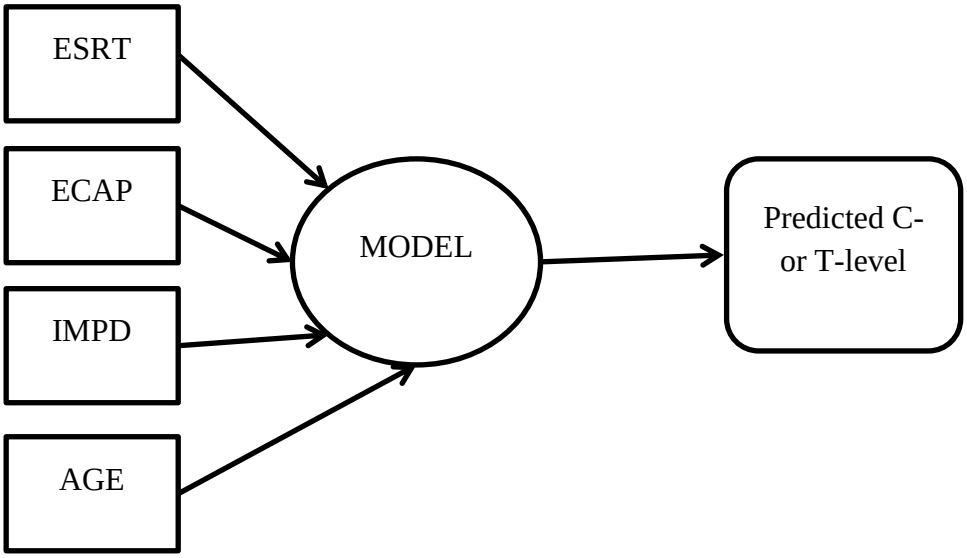

Figure 27 – Overview showing how the objective data (ESRT, ECAP, Impd and age) can be used to predict C- and T-levels

4.4.2 Splitting the data.

The dataset will be split into one training set and one test set. During training leave one out cross validation will be used to optimize parameters, choosing features and models. 20% of the data set will be removed before training and used as test set. These values will be removed before scaling and centring to avoid any information to be transferred from the test set to the model. During testing the test data be transformed using the same values as used on the training set. This means that the test data will be scaled and centred according to the standard deviation and mean value of the training set.

Section 3.4.4 discussed the pros of a three way data split: training, test and validation set. Since the data set is fairly small, taking 20% for a test set and 20% for a validation set will probably have a large negative impact on the end result. Instead of using this three way split, a method called *cross validation* will be used. There will still be set aside 20% of data dataset for testing, but the rest of the dataset will be kept in one group. Instead of having a fixed set for validation the training process will be repeated n times, where n is the number of samples in the training set. For each of the samples in the training set it will be removed from the training set, the model will then be trained on the n-1 other samples and then tested on the one left out. The performance is then the average over all the iterations. After some samples was removed as explained in 4.3.2, there are now 138 samples in the training set and 35 samples in the test set.

4.4.3 Choosing features

The input with the corresponding output has been analysed both using theoretical correlation methods, but also visually in plots. The correlations do not seem to be good enough to make a linear prediction based on one feature. It may be the case as discussed earlier that one feature does not contain any useful information alone, but in combination with another is very useful. Even if we do not know what features that are the best, it is still necessary to choose only a few. We could try all combinations and compare the results and take the best. Using this method we actually add a fair amount of additional complexity, as explained in 3.4.3 it is possible to overfit the validation data if we make a lot of decisions based on the results from testing on the validation set. There are other methods such as backward elimination and forward selection[52]. Backward elimination deletes each feature from the data set to see which one improves the accuracy the most when removed. This is done iteratively until only a few features are left. Forward selection does the exact opposite, adding each of the variables to see which ones improves the accuracy the most. This process is done iteratively until there is no further improvement from adding features. There exists also something called bidirectional elimination which is a sort of mixture of the two above.

Backward selection and forward selection can be fairly fast, while exhaustive search can be very time demanding depending on number of features. The large upside to exhaustive search is that it will find the global optima for the training set, while the other methods may get stuck in some local optima and we do not know how bad this local optima might be. Again it comes down to a trade-off situation, choose the risk of overfitting to have a greater chance of finding the global optima or a lower risk at overfitting, but a chance at being stuck in a bad local optima. It is often a good idea to choose robust options since overfitting is very common and can severely damage the performance.

Given the fact that we have to choose features for all 22 electrodes we could do an exhaustive search for each of them and then use the features that perform the best overall electrodes. This would make the search more robust since even if some noise causes the model to overfit with one feature, the same will probably not happen for all the other electrodes. This would then work as a regularization technique similar to how it is done in ensemble learning (see 3.3.4). By letting each electrode ‘vote’ for which features it prefers and the feature combination that get the most votes are chosen for everyone.

4.4.4 Error measure

Throughout the results in chapter 0 results from various results will be shown. To give a good intuition for the error three different errors measures will be used. Two of the model error measures will be the percent of errors below 5 and 10 current level (CL). This means the total number of errors that have an absolute value below 5 and 10 CL (see Figure 28). The reason for this is that about 5CL off the correct value is still so close that the patient may not notice the difference. Asking the patient twice can often lead to different values. This inaccuracy is caused by long programming sessions as explained in 2.2.3. This means if the error is less than 5CL we say it is a correct prediction, since a value that is so close that the patient has a hard time notice that it is a bit off is still good enough. Mean absolute error will also be used to see how the overall error of a model changes.

The error measures can be defines as follows:

Mean squared error (MSE) =

$$\frac{1}{n}\sum_{i=1}^{n} |(h(x_i) - Y_i)|$$

Threshold measure:

$$\frac{1}{n}\sum_{i=1}^{n} (|(h(x_i) - Y_i)| < threshold)$$

Where $h(x_i)$ is the output we get from testing the model on data x_i, from electrode i. Y_i is the correct value (T- or C-level) for electrode i. If nothing else is specified the error is measured using leave-one-out cross validation (see 4.4.2).

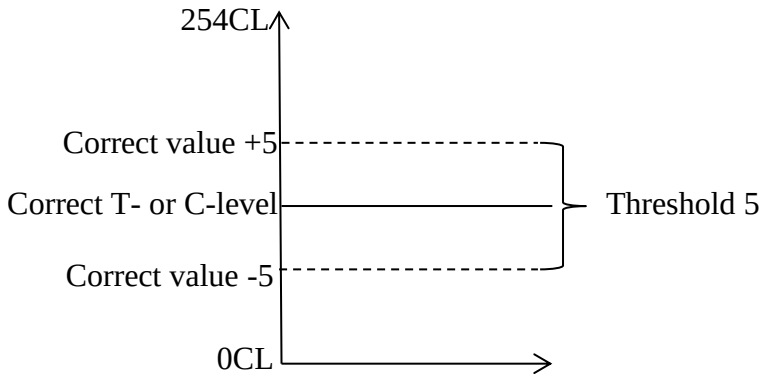

Figure 28 – Shows what area of errors that is covered by an error measure with threshold 5. The percent of error that lies in the area marked in the figure will be referred to as the percent of error below 5CL, or the accuracy of the model.

5 Experiments

This chapter will explain the various experiments that were performed to make a model that possibly predict C- and T-level of a Cochlear Implant speech processor and what steps were taken to improve the results. The experiments are divided into multiple sections that investigate possible solutions to predict the C- and T-levels as accurately as possible. Figure 29 shows an overview for this chapter and Table 1 shows more detailed what sort of models and data is used in each section. The first experiment in 5.2 investigates how various models perform using the objective data. In section 5.3 the features created in 4.3.3 are tested also with various models (see Table 1). Section 5.4 investigates how a model can be used to predict parameters in situations where some of the parameters are already known. Section 5.5 looks at the possibility to combine the results from 5.3 and 5.4 into one model, meaning how both objective measures and known parameters can be used in the same model. Section 5.6 use the results from all the earlier experiments to create a model that takes into consideration that a model, if used in practice needs to be very robust. There will be a short discussion for each result in each section, but section 5.7 discusses the overall result and some pros and cons with some of the best models. In section 5.8 a final model is tested on the test set.

All the results came from using cross-validation as explained in 4.4.2. There will be mainly three error measures used in each experiment, percent of error below 5CL, percent of error below 10CL and mean absolute error (see 4.4.4). When discussing the results the percent of errors below 5CL will be referred to as the *accuracy* of the model. As explained in 4.4.4 errors less than 5CL is a very low error, and we may call that a correct value. The accuracy of the model will also be the main focus throughout the experiments.

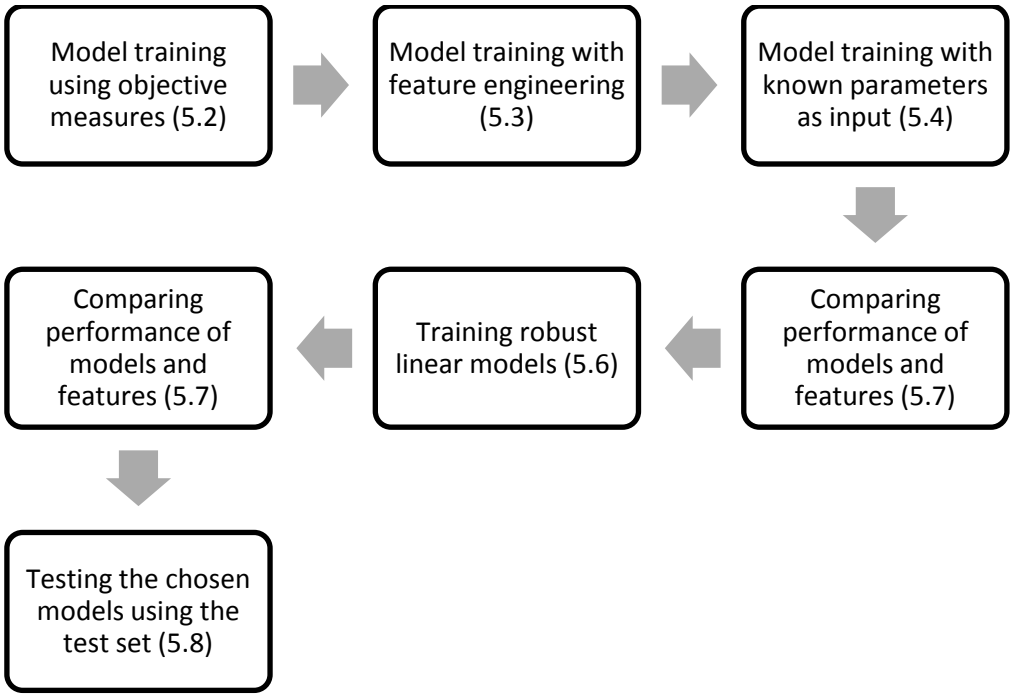

Figure 29 - flow chart for what sort of experiments are performed in each section.

Section	Number of known parameters	Model	Features
5.2	None	Linear Regression, SVM(RBF) and ANN	Objective data (ESRT, ECAP, Impd and Age)
5.3	None	Linear Regression, SVM(RBF)	Objective data and features from 4.3.3
5.4	1 and 2	Linear Regression and SVM(RBF)	None
5.5	1 and 2	Linear Regression, SVM(RBF) and ANN	Objective data and features from 4.3.3
5.6	1 and 2	SVM(linear) and boosting(linear regression)	Objective data and features from 4.3.3

Table 1 – Shows what sort of features and models that are used in each section.

Constant function

In the various experiments the error of a constant function will be listed together with the performance of the trained models. A constant function means that we always predict the same value no matter which sample we see and what the data might be. The predicted value is the mean of all the C- or T-levels for that electrode. This is the same value we get if the sum of square error is minimized a constant input which gives a constant output. This is also the same model we get if a k nearest neighbour algorithm is used where k is equal to the number of samples. Listing the constant function makes it easier to see how well the model has performed. Since the constant function is the simplest model possible it should be possible to get better results with more complex models, but if the results are worse than a constant function we know that the model has overfitted.

5.1 Models used

Section 4.2.1 discussed some aspects of why it is important to explore and understand the data to increase the chance of success when applying a prediction model. Section 4.2.2 analysed the data and showed that the data have poor correlation to C- and T-levels. As discussed this implies that linear models will probably not work very well with the objective data. However, from the analysis we cannot say that a non-linear model will perform any better. Since we do not know what sort of model is likely to perform best a few different models will be used. As there is a large amount of work concerning implementing an algorithm and tuning it for a specific problem, there is no time to try out "all" algorithms. This chapter will therefore focus on a few well known and commonly used algorithms and use cross-validation to evaluate and compare their performance

There are mainly three methods used in the experiments, linear regression, SVM and ANN. Linear regression is used from MATLAB Curve Fitting Toolbox™, ANN is used from MATLAB Neural Network Toolbox™ and SVM is used from LIBSVM[53]. The experiments will use variations of linear regression and SVM, ANN will be used where linear regression and SVM is not sufficient or does not have some capabilities that ANN has. All these models and their training function were explained in 3.3.

5.2 Prediction with objective measure

The first experiments are done with simple models in order to better understand the problem and to gain some intuition for what kind of features are actually useful making a prediction. After that more advanced models are used to investigate how complex the model is required to be to fit the data well.

5.2.1 Results using linear model

Results from using the objective measures to predict T- and C-levels with linear regression can be seen in Figure 30 and Figure 31, respectively. It is however not clear from the results what features are the most useful or performed best. They all seem to add some information but using all four features does not seem to be the best option. The best accuracy is from using ESRT and Impd together for C-levels and ESRT for T-levels. Using ESRT and Impd for C-levels gives 17.2% accuracy. Compared to a constant function which gives 11.2% this is an improvement of 6%. The best result from T-levels gives 18.4% accuracy, compared to a constant function which gives 16.5% for T-levels this is an improvement of 1.9%. This tells us that the objective measures are more useful for predicting C-levels than T-levels.

The errors we get using the linear regression is not at all that impressive, the best features give an accuracy of 17.2%, meaning that the algorithm fail on 82.8% of all the patients when

predicting C-levels. Similar with T-levels, accuracy of 18.4% corresponds to failure on 81.6% of the electrodes. This is not unexpected since the data showed low correlation to the C-/T-levels in 4.2.4.

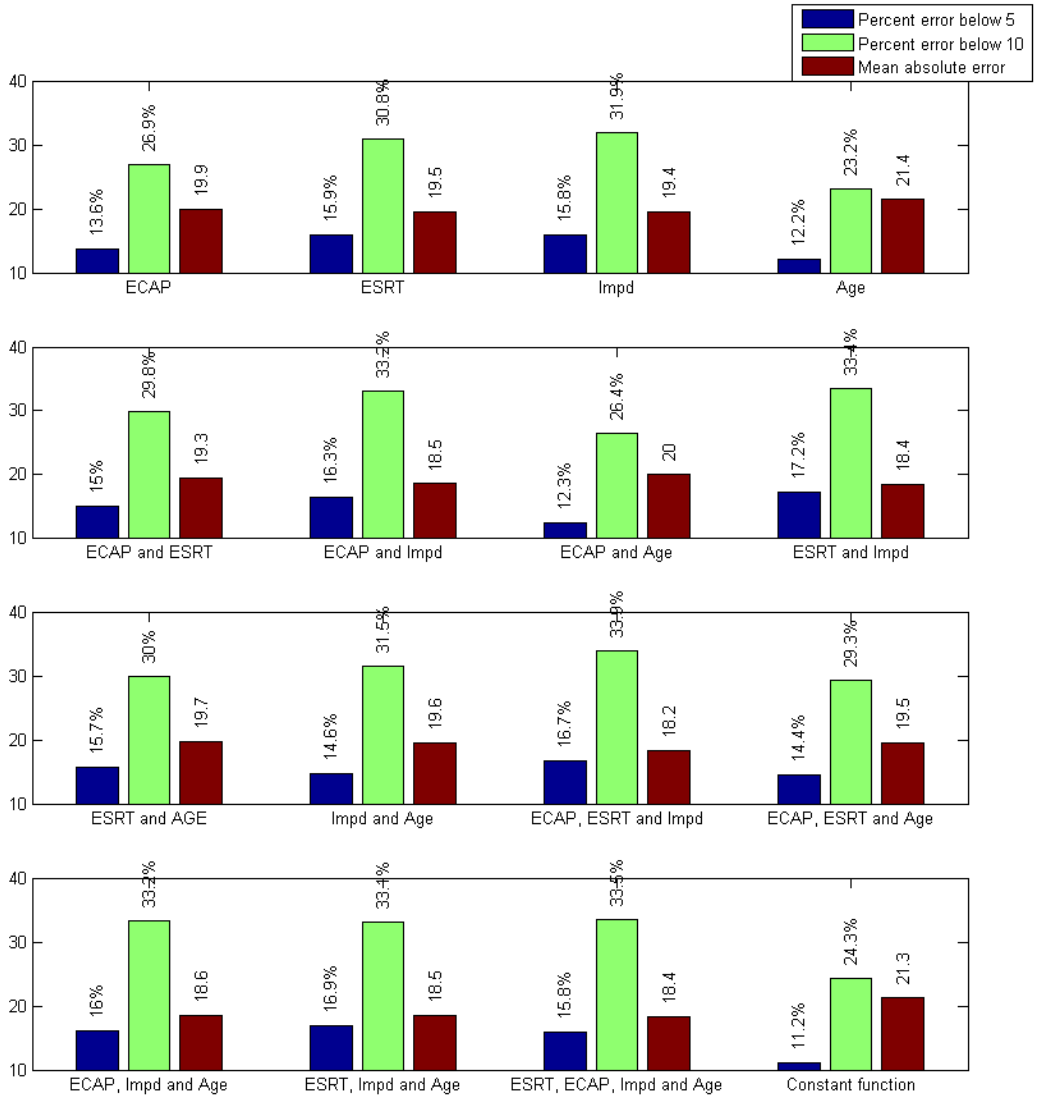

Figure 30 – Shows cross-validation error when predicting C-levels for all combinations of features. What features that were used are listed under the bars. All models were created using linear regression.

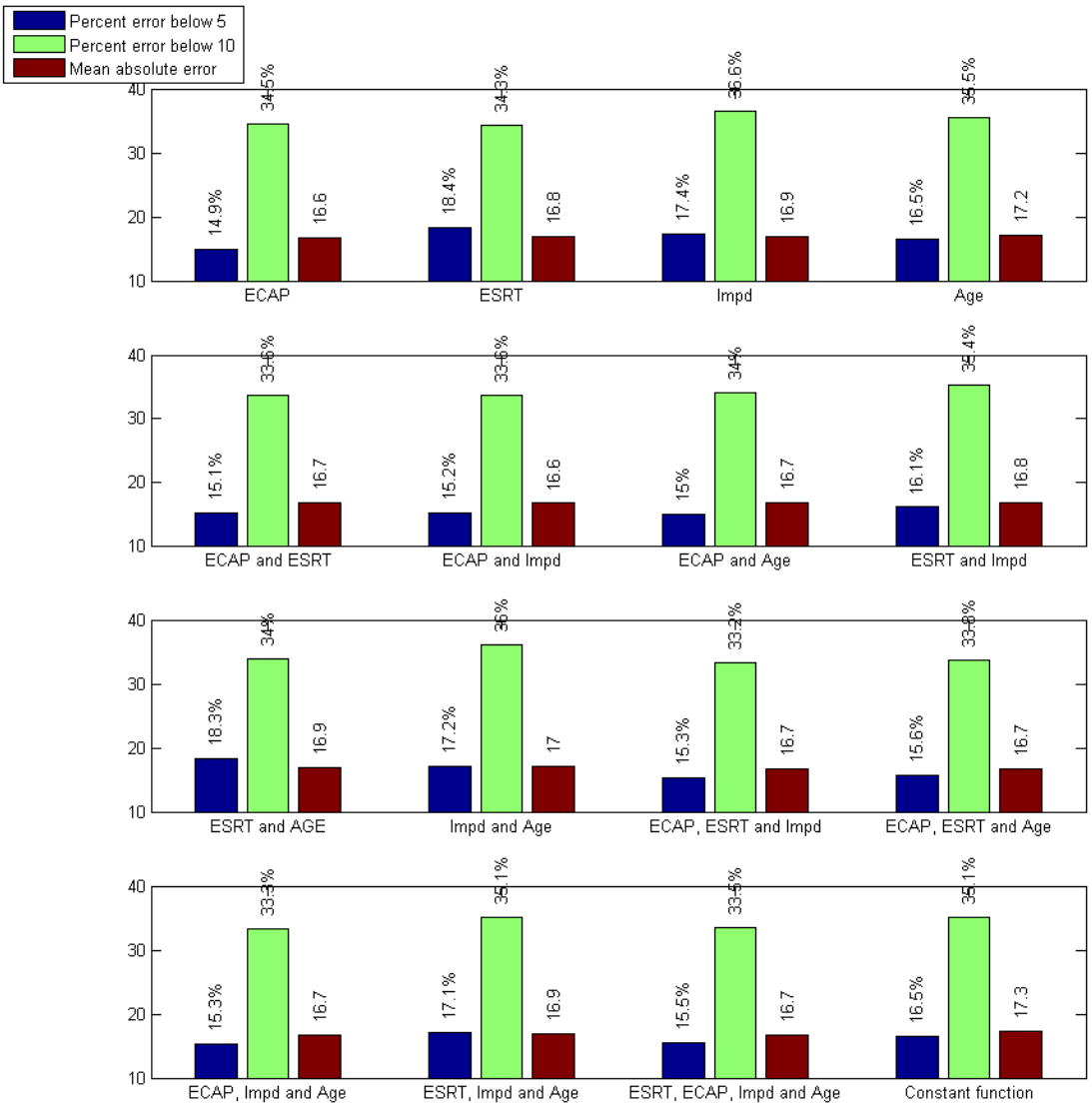

Figure 31- cross-validation error for combination of features, using linear regression for predicting T-levels

This experiment used only linear regression, but as discussed in 3.8 it is not clear how complex hypothesis is needed to make accurate predictions. Before testing out the features discussed in 4.3.3, more advanced models should be used to make sure the linear model is not underfitting the data. If the results improve when using more complex models, it is likely that the linear model is not powerful enough to model the data.

5.2.2 Results from ANN and SVM

As mentioned earlier the correlation analysis takes only linear correlation into consideration, which means that non-linear models can perform very well even when the data is poorly correlated. This section will test some models that have functionality for approximating non-

linear functions. This will hopefully give some indication whether a non-linear model is able to get better results.

The models tested here is ANN and SVM. The ANN use Levenberg-Marquardt backpropagation training algorithm (explained in 3.3.2). The network has one input neuron for each feature, three neurons in the hidden layer and one output neuron for C- or T-level. The size of the hidden layer is chosen to be fairly small because the goal is for now to see if the algorithm will improve with a non-linear model. Setting the number of neurons small will decrease the chance for the model to overfit, but will still be able to approximate more complex function than a linear model. The SVM used is a SVM regression model with an epsilon loss function[53] (explained in 3.3.3). The kernel is radial basis function, which is chosen based on its capabilities to solve a variety of problems and for its non-linear capabilities (see 3.3.3).

The C (cost) and gamma of the radial basis function is a bit tricky to set, since we are training so many models with many different features. The C parameter for the SVM should not be confused with the C-level parameter for the CI, C and gamma was explained 3.3.3. The parameters will be set based on exhaustive grid search for these two parameters. To keep things simple and decrease the chance of overfitting the parameters that perform best overall feature combinations will be used for all features and all electrodes. This may give some features an unfear disadvantage, but finding the optimal parameters for each electrode and features takes a very long time and makes the model more vulnerable to overfitting. Either way the goal is for now not to get the very best results possible, but to get an estimate for how the model might perform.

The results from running the parameter grid search for T-level can be seen in Figure 32. We can see that having large gamma and large C is probably not a good idea (bottom right corner). It seems however that the models with best performs are the ones with the lowest C and gamma. In Figure 33 is the same result with C-levels, which shows similar results except from the upper left corner. It seems with C-level it is a good idea to have higher values for both C and gamma. A gamma of about 0.0625 seems like a good option, C should however be as low as possible since it controls the models complexity, so a C value of 25 seems fine for predicting C-levels. The best parameters for prediction T-levels seems to be with C equal to 5 and gamma of 0.001.

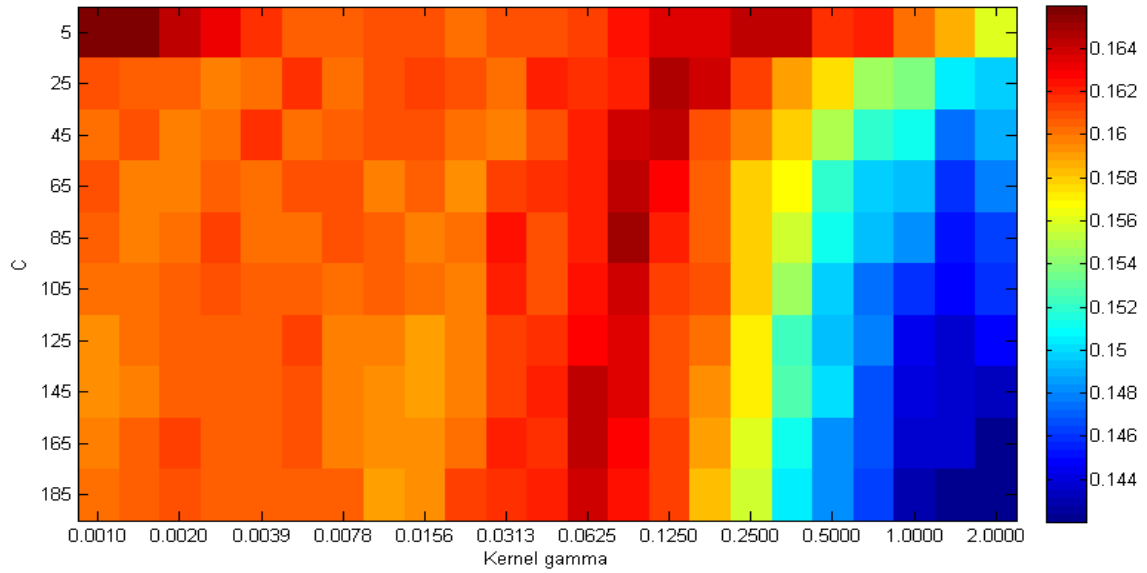

Figure 32 – Illustrates how the average performance of models trained with all combinations of features change as the SVM parameters C and the kernel gamma change. The value in the colour map is the average percent of errors below 5 when predicting T-level.

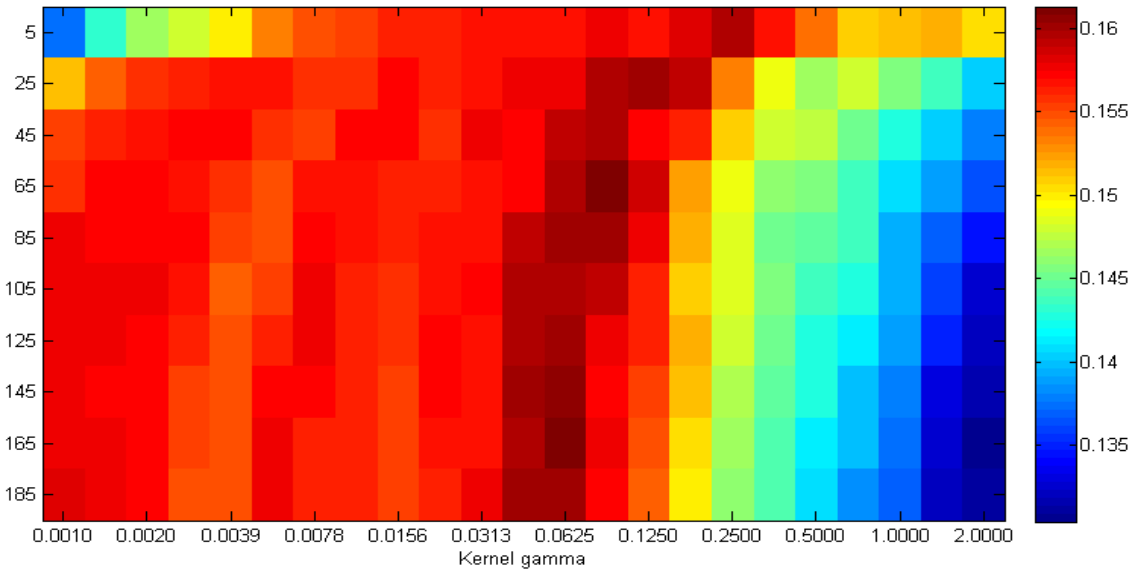

Figure 33– Illustrates how the average performance of models trained with all combinations of features change as the SVM parameters C and the kernel gamma change. The value in the colour map is the average percent of errors below 5 when predicting C-level

C-level results

Testing SVM and ANN in the exactly same way as with linear regression gives the results in Figure 34 and Figure 35. It is interesting that ECAP is one of the features that give highest error for all the models, when used alone. When comparing these results from ANN and SVM with the results from linear regression we can see that the error using only age is lower with the non-linear models. This means that age probably has a non-linear relationship with C-levels that a linear model is not able to utilize. It is also interesting that using multiple features is not necessary better than using just one or two. Overall the best results from using these non-linear models were not much better than the results from using linear models, especially the neural network seems to overfit the data. The best result from predicting C-levels with linear regression where 17.2%, compared to 18.8% using SVM, this is an improvement of 1.6% when using non-linear model compared to linear model for predicting C-levels.

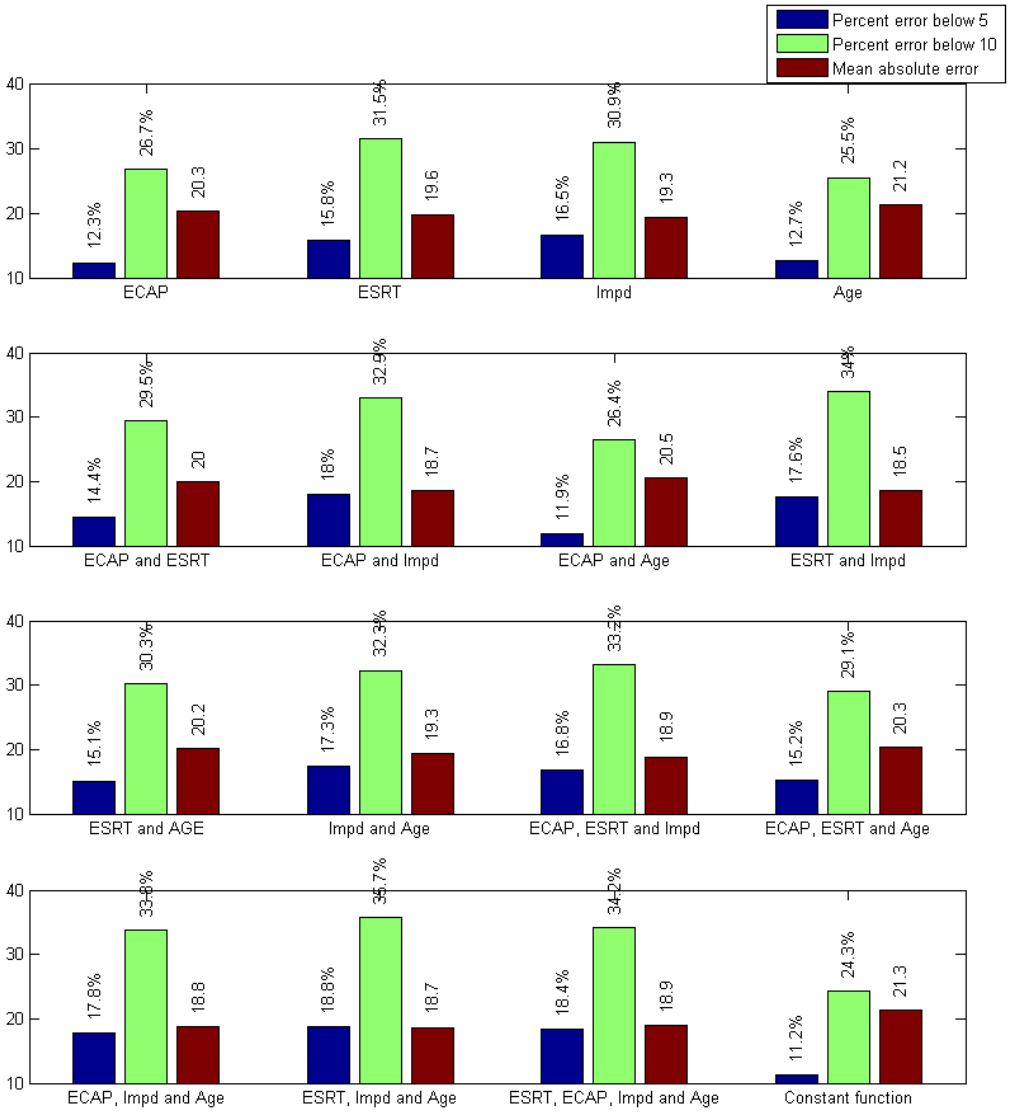

Figure 34- Cross-validation error for combination of features, using SVM for predicting C-levels

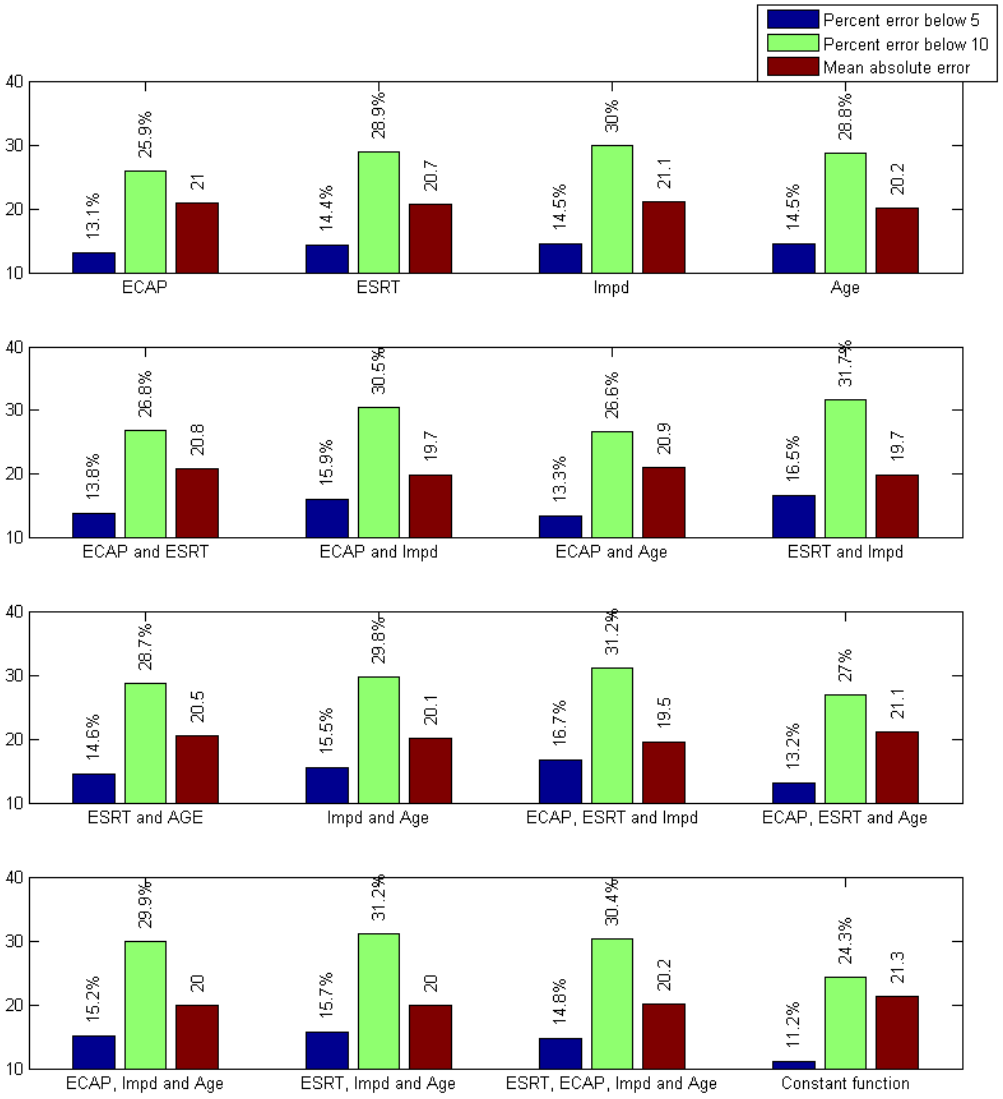

Figure 35 - Cross-validation error for combination of features, using ANN for predicting C-levels

T-level results

Using non-linear models to predict T-levels does not seem to give much better results than with a linear model. Most combinations of features do not give results better than not using any data (see Figure 36 and Figure 37). The best results we get when using only ESRT with SVM, which gives an accuracy of 18.4%. As with C-levels the ANN does not seem to perform very well, it seems like it may be overfitting the data since some results give worse accuracy than the constant function.

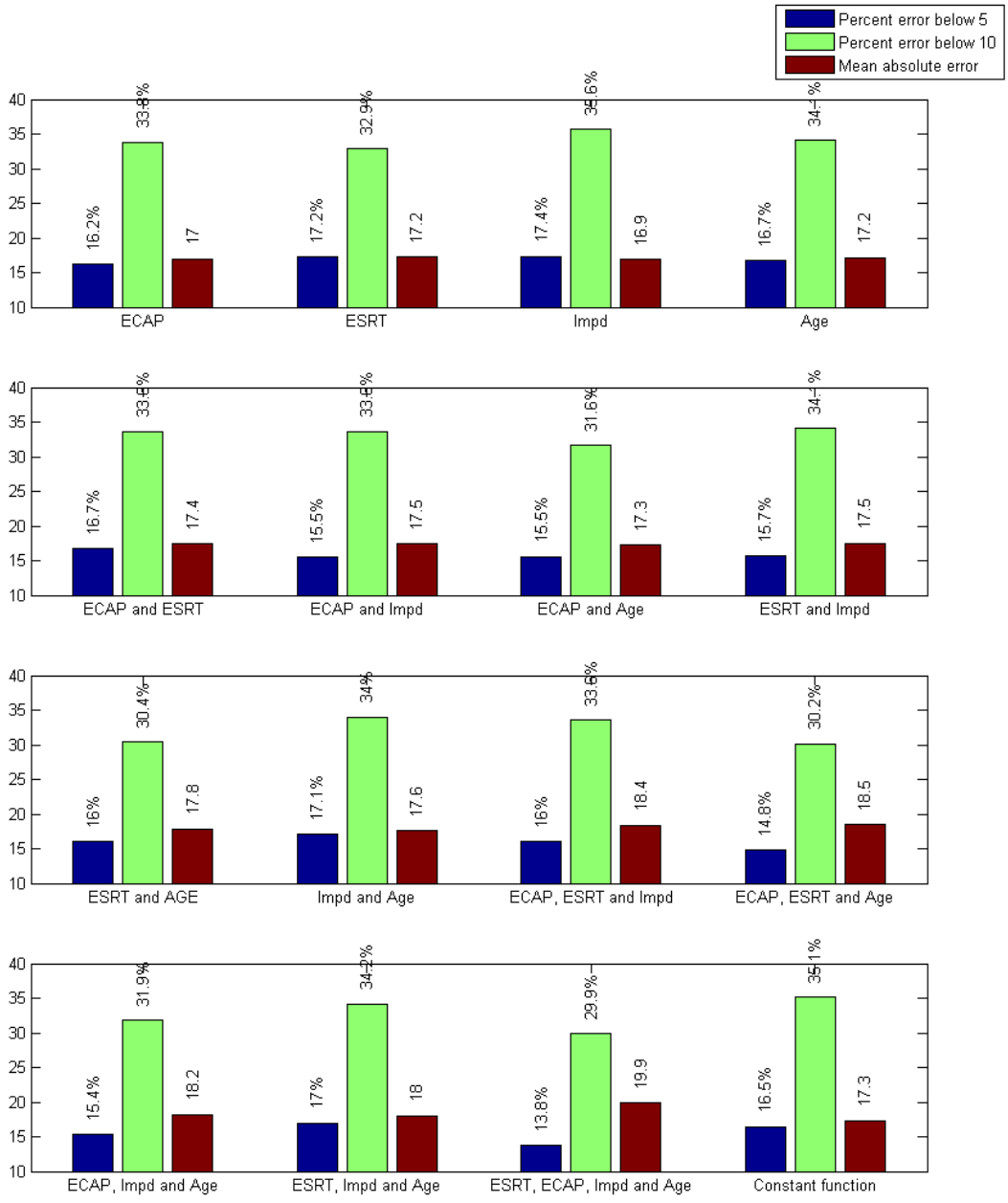

Figure 36- Cross-validation error for combination of features, using ANN for predicting T-levels

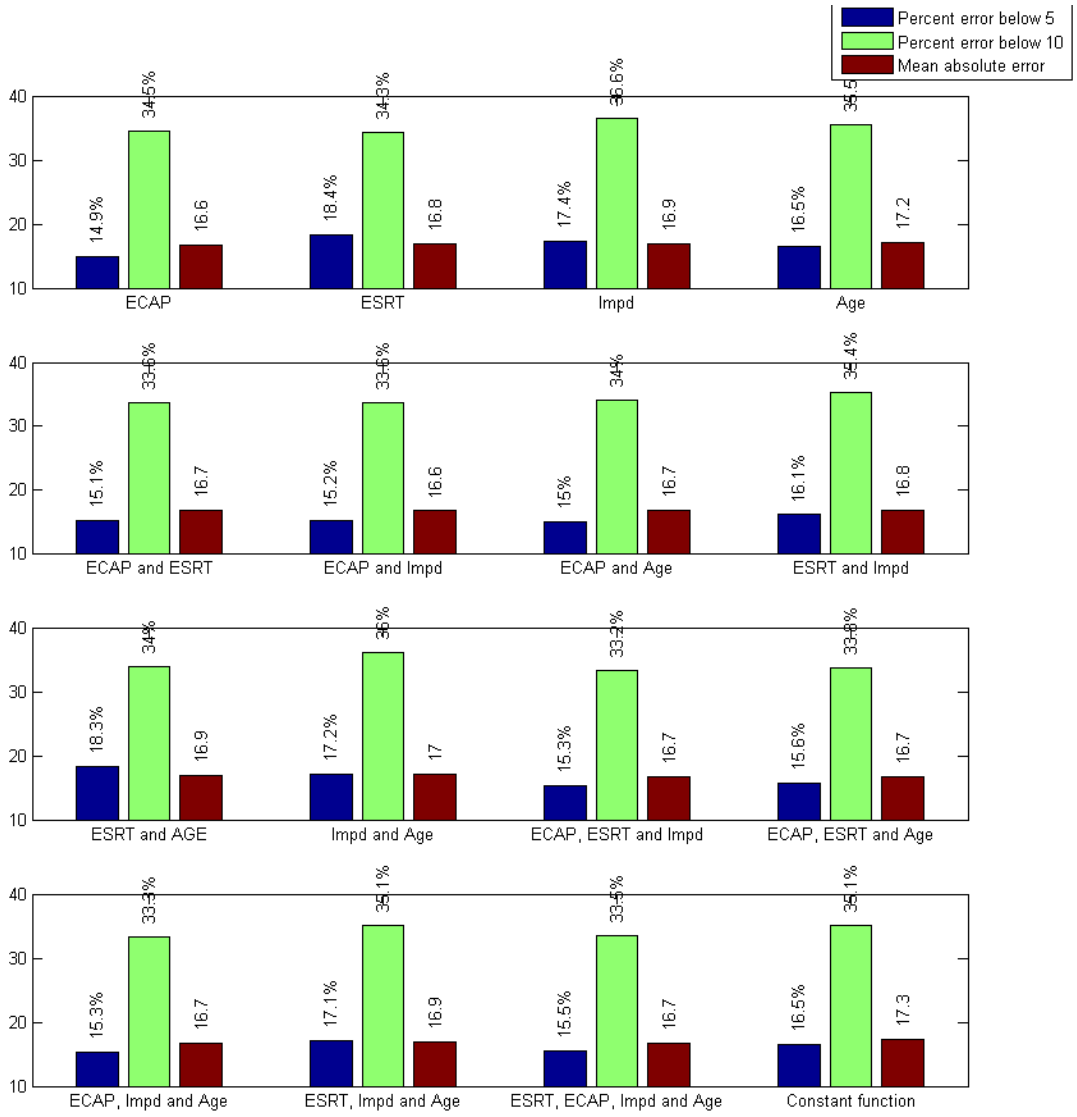

Figure 37 - cross-validation error for combination of features, using SVM for predicting T-levels

One of the reasons for trying out ANN and SVM was to see if the accuracy did increase and see if the linear model was suffering from underfitting. The results did however not improve and we can from these results not conclude that the linear model is suffering from underfitting alone. There are many other possible reasons that the error is not improving, underfitting is still possible, but it could also be that the data set is too small or the data is not good enough (e.g. too noisy). As previously discussed getting more data is not easy and not possible at this time, so the only option seems to either improve the data somehow or adjust the models.

5.3 Prediction with feature engineering and feature selection

A few possible shortcomings with the data were discussed in 4.1.2 and 4.3.3. Section 4.3.3 proposed two possible ways to extract features from the objective measures. Each of them can be applied to ESRT, ECAP and impedance, which gives six new features. These six new features will be treated like the objective measures. They will also be scaled and centred as explained in 4.3.4. For the rest of the experiments these new features will be referred to as f1 and f2. The feature f1 use PCA to reduce noise and redundant information, f2 create a bias corrected value of an objective measures. An objective measure based on f1 or f2 will be written as the name of objective measure followed by an underscore and the name of the feature. E.g. feature1 applied to impedance will be written as Impd_f1.

In this next section these new features will be tested against the other features. Hopefully these features will be easier for the models to use and maybe improve the accuracy of the model. Because the number of features are now fairly large, the feature selection methods discussed in 4.4.3 will be used instead of listing all combinations. As before linear regression will be tested before more advanced models are tested.

Results C-level

The results from running the same experiments as earlier only with all the features give the results in Figure 38. The best result gives 18.47% accuracy, with features impedance ECAP_f1 and Impd_f2 or ECAP, ECAP_f1 and impd_f1 with a linear model. The best results when using SVM is with features Impd, Ecap_f1 and Impd_f2. Using the new features actually seems to improve the results, although it is a bit strange that the mean absolute error did not improve. It is also interesting that impedance is chosen very often, also multiple features based on impedance are often chosen. Age is the only feature that is not used among the top 5 results with linear regression, which is not surprising because it has earlier (5.2.1) given poor results for linear models. Age is however used in the SVM model which makes the evidence that age requires a non-linear model even stronger. When using these new features we achieved on accuracy of 18.5% using linear regression, compared to the results when not using these features this is an improvement of 1.3%.

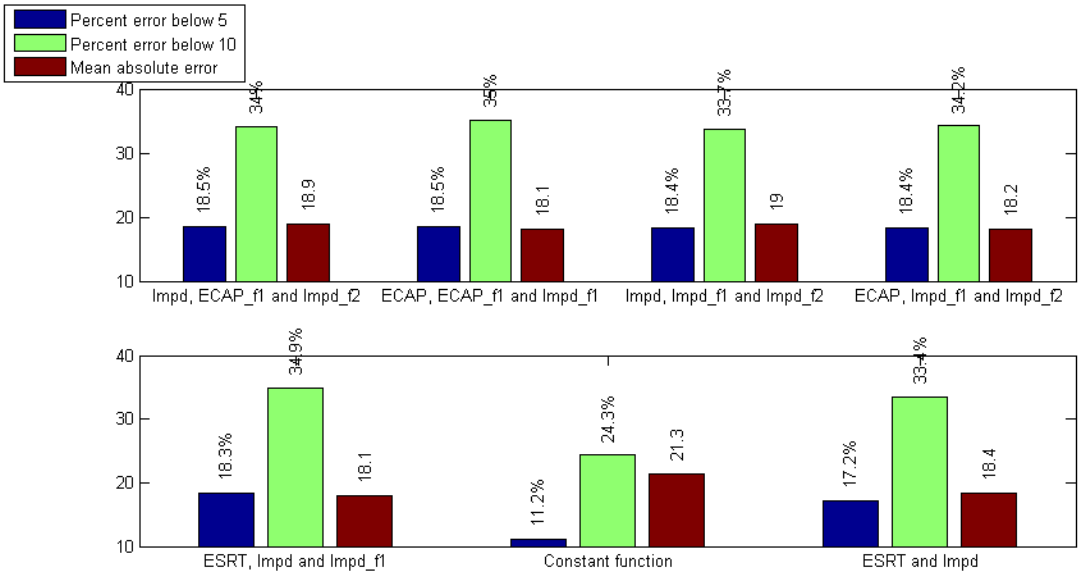

Figure 38- Top 5 results from predicting C-levels with combinations of features on a linear model. Constant function and ESRT and Impd are listed for comparison. ESRT and Impd together gave the best results of all the results without the new features.

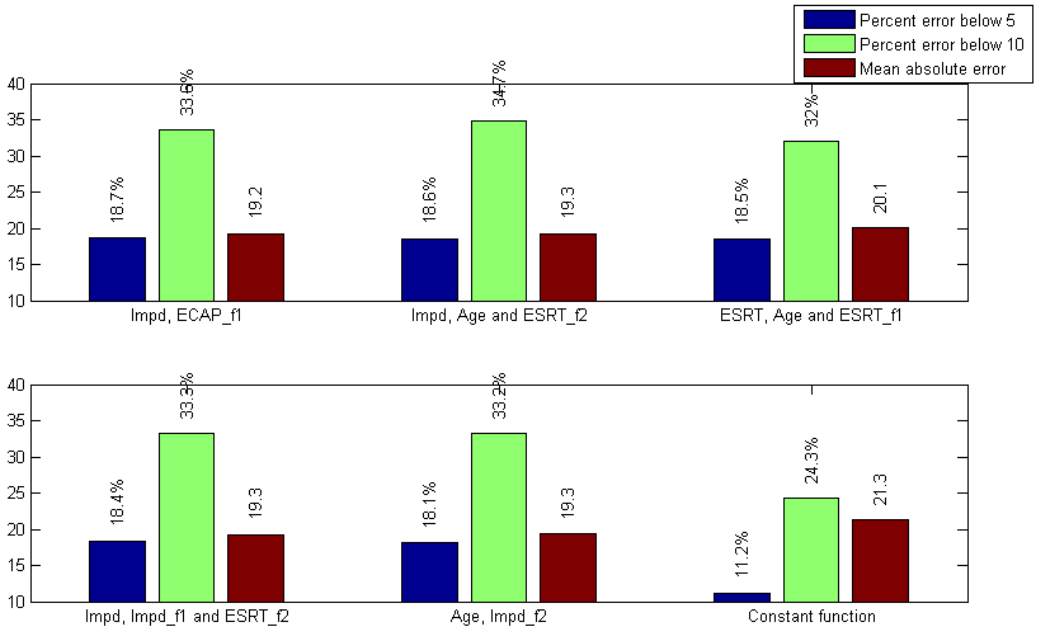

Figure 39- Top 5 results from predicting C-levels with combinations of features on a SVM. Constant function and ESRT and Impd are listed for comparison. ESRT and Impd together gave the best results of all the results without the new features.

Results T-level

The results from predicting T-levels give fairly similar results as those we saw for C-levels. Using the linear model gave with these new feature gives almost exactly the same result as without them (see Figure 40). Using only ESRT in a linear model we get an error of 18.4% which is the same as with the new features. It is still a few percent better than using a constant function. Using the SVM (see Figure 41) the results got a bit better, the best results was achieved with Impd and Impd_f1, which gave an accuracy of 20%.

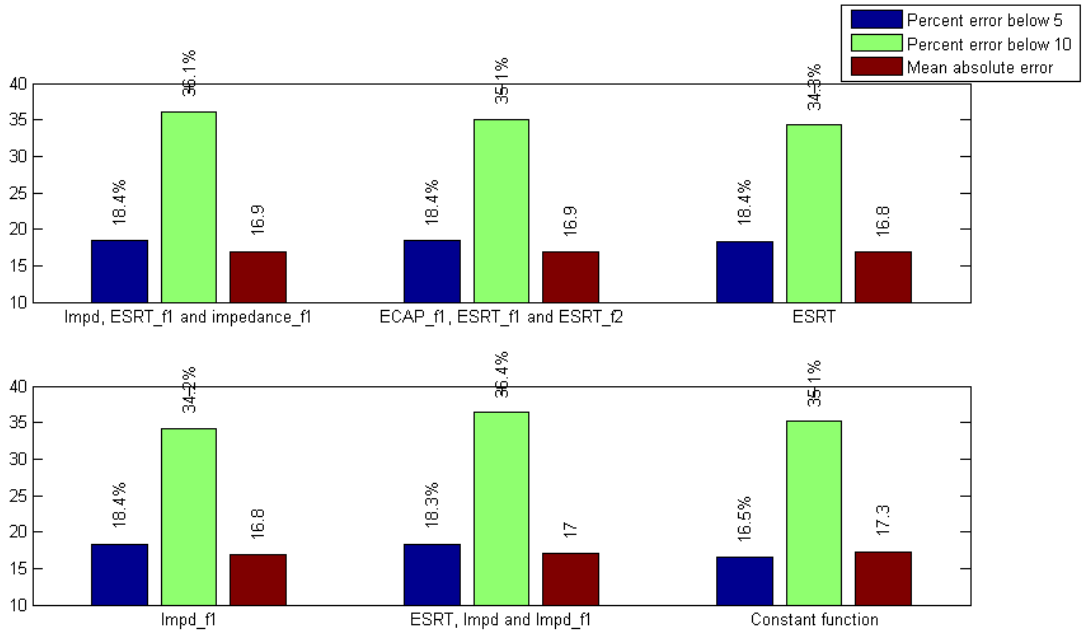

Figure 40- Top 5 results from predicting T-levels with combinations of features on a linear model. The constant function is listed for comparison.

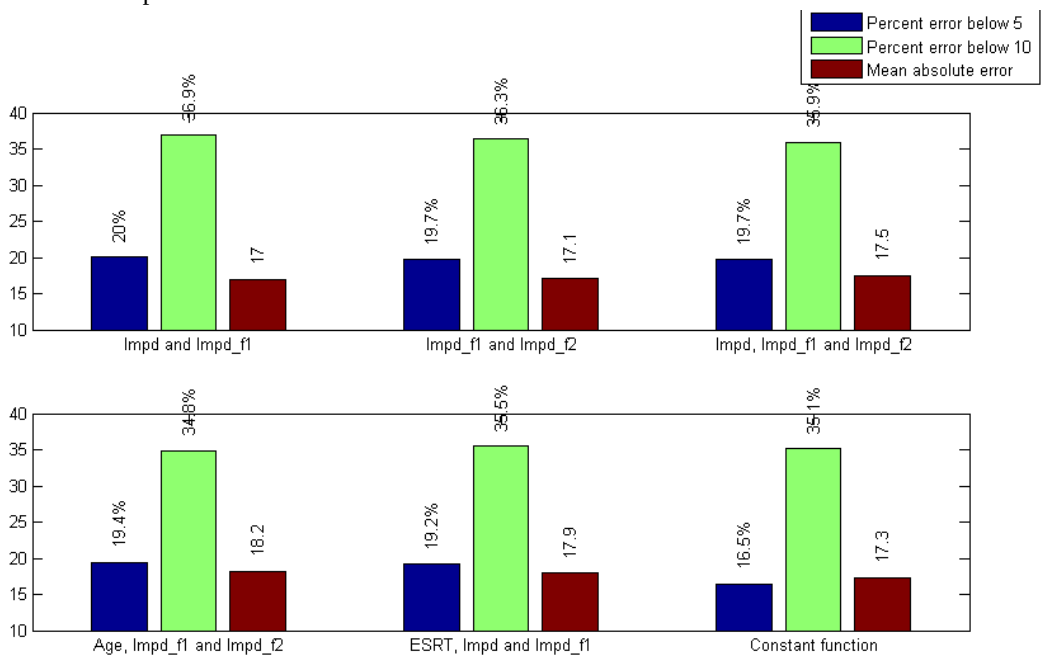

Figure 41- Top 5 results from predicting T-levels with combinations of features on a SVM. The constant function is listed for comparison.

5.4 Prediction with known parameters

After trying a wide variety of models and features, some showing better results than other, but still the errors seem too large to be useful to automatically predict parameters. If the model is going to be useful in practice it needs to be fairly accurate for most patients. The low accuracy seems to either be caused by not enough training data or that the data is simply does not have a strong enough relation to the C- and T-levels. Both these problems are not easy to solve, but maybe the problem can be changed a bit.

As discussed earlier neighbouring electrodes have often quite similar parameters. As explained in 2.2.3 one of the main problems when fitting a CI is that there are so many electrodes that need to be programmed individually. Finding the parameters for one electrode is quite fast, but doing it 22 times is more demanding. With this in mind maybe we can let the audiologist find a few parameters and then let a model use these values as input to predict parameters for the rest of the electrodes. The algorithm would then be able to use these known parameters as a sort of baseline for unknown parameters.

This section will investigate the possibility to create a model that can predict C- and T-levels when a few of the parameters are already known. This section will not use the objective data, combining objective data and known parameters will be discussed in 5.5.

5.4.1 Measuring performance

In the previous experiments the errors have been measured as mean absolute error and percent of errors below a certain threshold. The values were compared to a constant function in order to tell if the algorithm actually had learned something from the data. When using one of the electrodes as input it does not make sense anymore to compare the results to a constant function. Since the model is now going to use known parameters as input, comparing the model to a constant function will no longer be a good indication for how much the model have learned. For a function to be comparable we want to use the same input as our algorithm without actually using data from other patients. This way we can see if learning from previous patients is beneficial or if using the known values alone is just as good. One such function could simply interpolate the known electrodes and return the value for the unknown electrode that we want. If the input is only one electrode this is a constant function, but for multiple electrodes as input this function can return the interpolated (and extrapolated) values using all the known electrodes. To keep this function simple and robust linear interpolation will be used; this is also used sometimes in practice. When extrapolating nearest neighbour will be used, both extrapolating with a linear function and with nearest neighbour was tested using cross-validation. Nearest neighbour gave best results and will therefore be used in order to give the fairest comparison.

5.4.2 Finding optimal electrodes

It is likely that some of the electrodes are more useful to predict the unknown electrode parameters than others. It is also desirable to minimize the number of electrodes that are needed for the model to make accurate predictions. We are therefore in a situation with a trade-off between as good prediction as possible and a small number of required electrodes as possible. The goal is to find what electrodes are the best electrodes for the model when the audiologist is able to find some number of electrodes. The number of parameters the audiologist is able to find depends on the patient, so a model that can handle a variable number of known values would be best.

In order to find the optimal electrodes a selection method similar to the other experiments will be used. Various combinations of electrodes using cross-validation will be tested, which will be compared and the electrodes with the lowest error will be used in later experiments. The electrode that is used as input is not tested since the model will get them perfectly and it will only make the results look better.

5.4.3 One known parameter as input

The experiments here will focus on only linear regression since we are first interested in finding which electrode parameters are most useful for predicting unknown parameters. More advanced models will be used in 5.5 and 5.6.

Results C-level

As shown in Figure 42 and Figure 43 the accuracy improved significantly by using one known C-levels as input to the model. It is also quite clear from the plot that using one of the electrodes in the middle gives the best results, while the ones at the end are not so good. There are some variations in middle as well and the electrode with highest accuracy is actually electrode 15. It makes sense that the electrodes in the middle is better than those at the ends because as the correlation analysis in 4.2.3 showed, close by electrodes have more similar values than values further apart. Using electrodes in the middle as input will then minimize the total distance between the electrode we have values for and the electrodes we do not have values for. From Figure 42 we can see that the accuracy is better with linear regression than a constant function. Using a linear regression model with one known parameters it seems that about 61% of all patients will get good C-level. If we take the best score from the linear model and the best score from constant function, we get a difference of about 2.5%. This means that 2.5% more of the total population gets a good programming when using data from other patients.

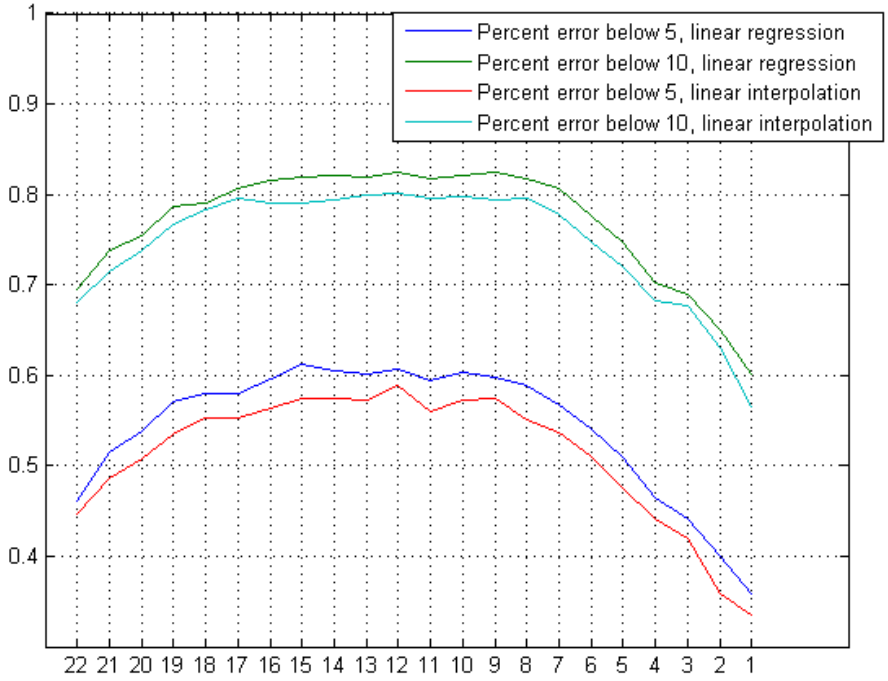

Figure 42 – Errors when using one electrode at a time as input to predict C-levels with a linear model. The x-axis is the input electrode used to create the model, while the y-axis is the error when using it as input.

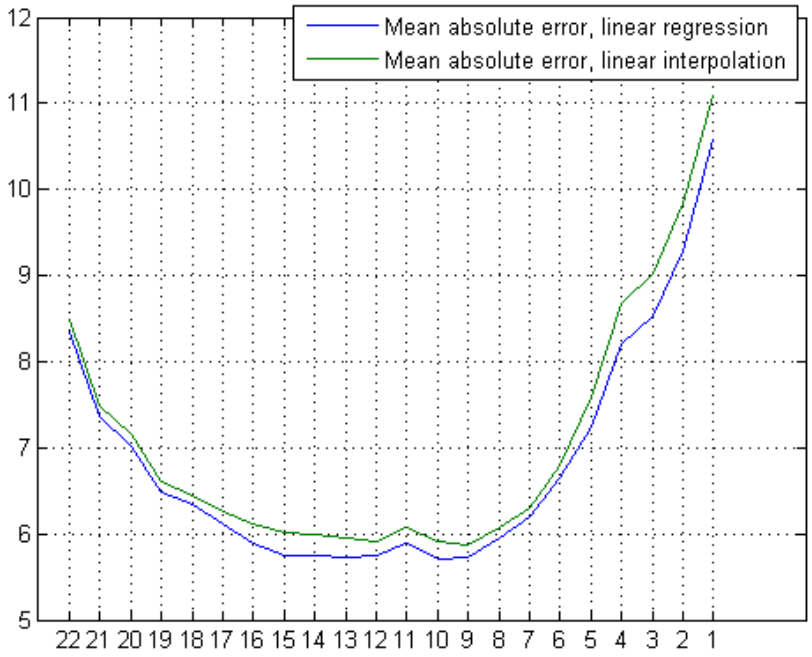

Figure 43- Plot showing errors when using each of the electrodes as input to predict C-levels with a linear model. The x-axis is the input electrode used to create the model, while the y-axis is the error when using it as input.

T-level

The same experiment performed on T-levels give similar performance to those from C-levels, but slightly better (see Figure 44). This can be explained by the lower variance in T-levels (see 4.2.2). It seems that in general taking one of the electrodes in the middle is the best options. Taking electrodes 11 seems to give slightly worse predictions than other electrodes in the middle. Electrode 9 might seem like the best option for T-level. Using electrode 9 to predict T-levels for all the other electrodes give an accuracy of about 66%, which is an improvement of about 3.5% over a constant function.

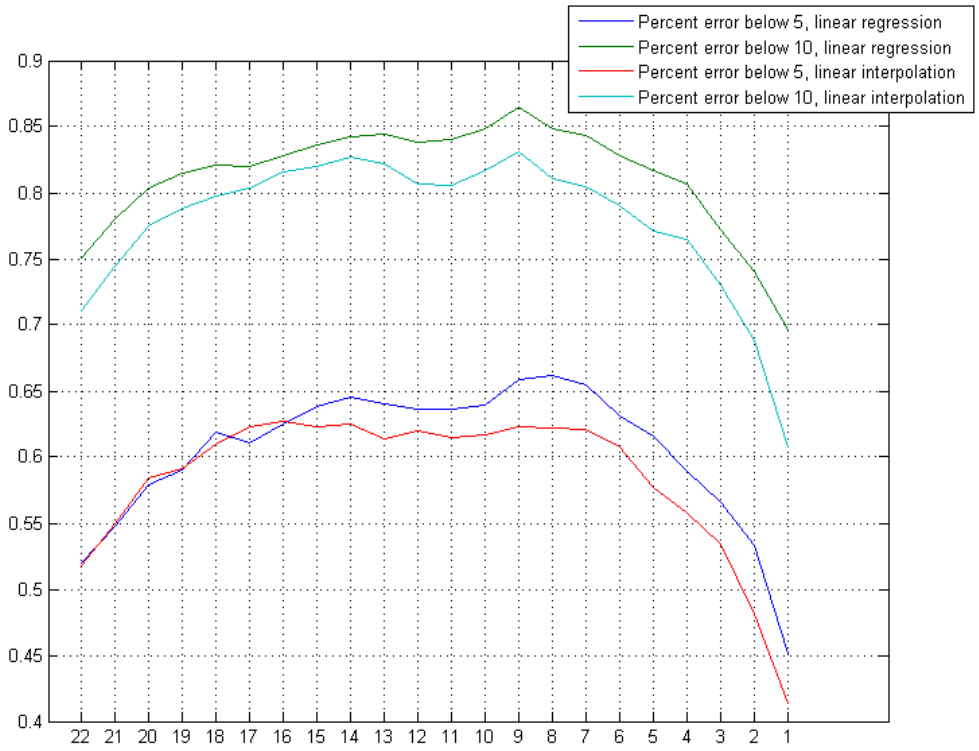

Figure 44 - Errors when using one electrode at a time as input to predict T-levels. The x-axis is the input electrode used to create the model, while the y-axis is the error when using it as input.

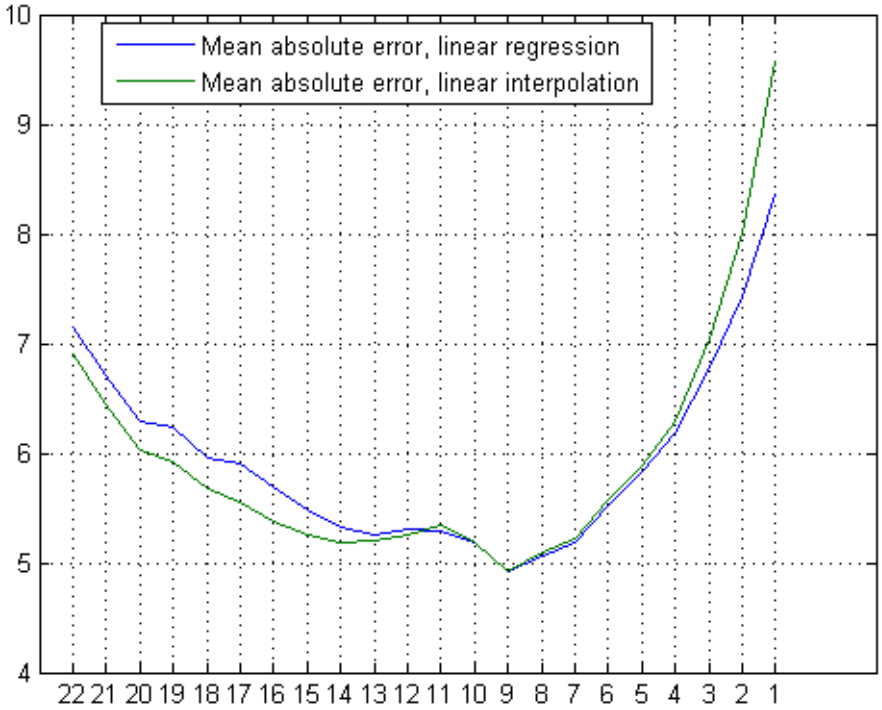

Figure 45 Errors when using one electrode at a time as input to predict T-levels. The x-axis is the input electrode used to create the model, while the y-axis is the error when using it as input.

5.4.4 Two known parameter as input

Using one known parameter as input seemed to reduce the error quite a lot, but using a prediction model did not seem to be a large improvement over a simple constant function. This section will see how a prediction model performs with two known parameters as input.

Results C-level

Figure 46 shows the accuracy of all pairs of known C-levels when using linear regression. The electrode combination that seems to give the best result is one electrode that is close to each end. The reason for this is probably the same as for one electrode, choosing the electrodes on the sides will minimize the total distance between known electrodes an unknown ones. The highest accuracy is achieved with electrodes 15 and 5, which gives 78.8%, see Figure 47. The number of errors below 10 is also pretty good. More than 90% of all the patients will get a value that is fairly good.

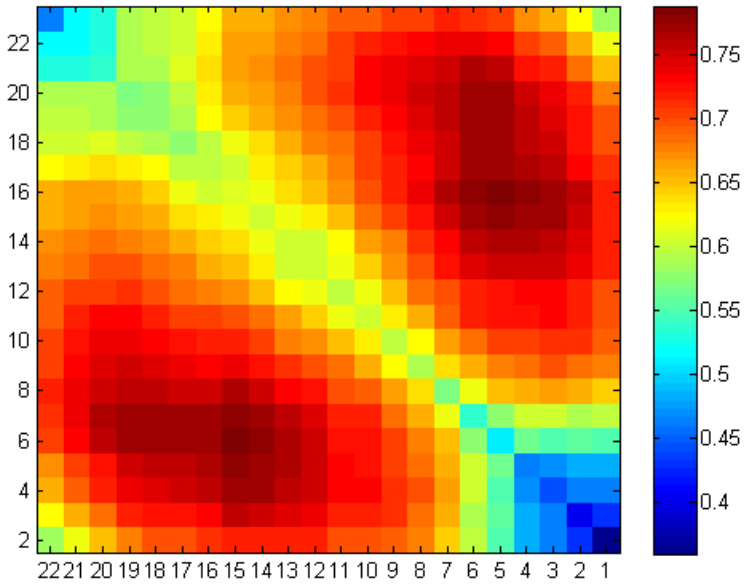

Figure 46 – Accuracy of all combinations of models using two known electrodes to predict C-levels.

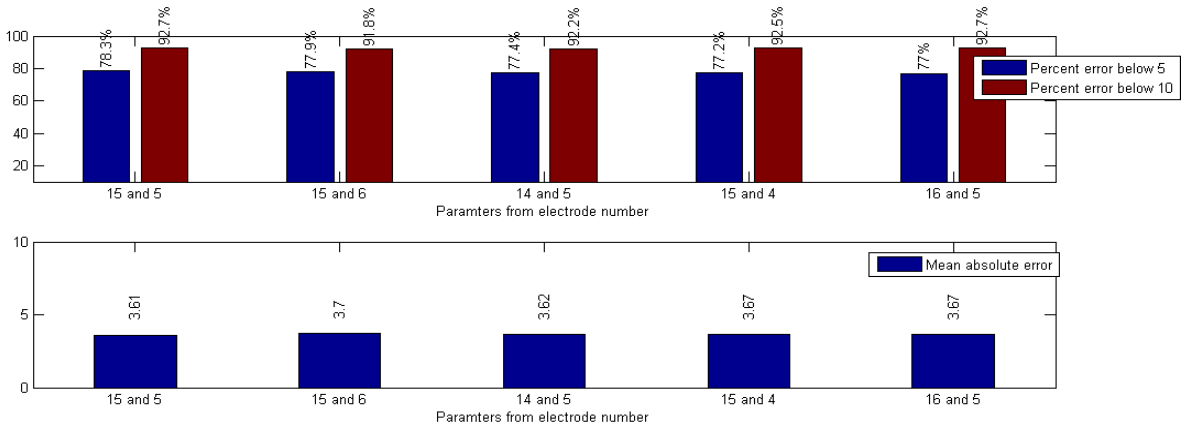

Figure 47 – Top 5 results from using two known parameters when predicting C-levels with a linear model.

Linear regression gave good results, but maybe a non-linear model will be even better. Since we now have a very different problem compared to section 5.2.2 where only the objective measures was used, a new parameter search for the SVM is required. However, to limit the time required to do this search only the top 20 best combinations of known parameters for linear regression will be used. This seems reasonable since it is likely that about the same known values are useful in SVM. The results from this grid search can be seen in Figure 48. The search shows that a higher gamma and C parameter is required now compared to when only the objective measures was used. A gamma less than about 0.01 and a C above 120 seem to give best results.

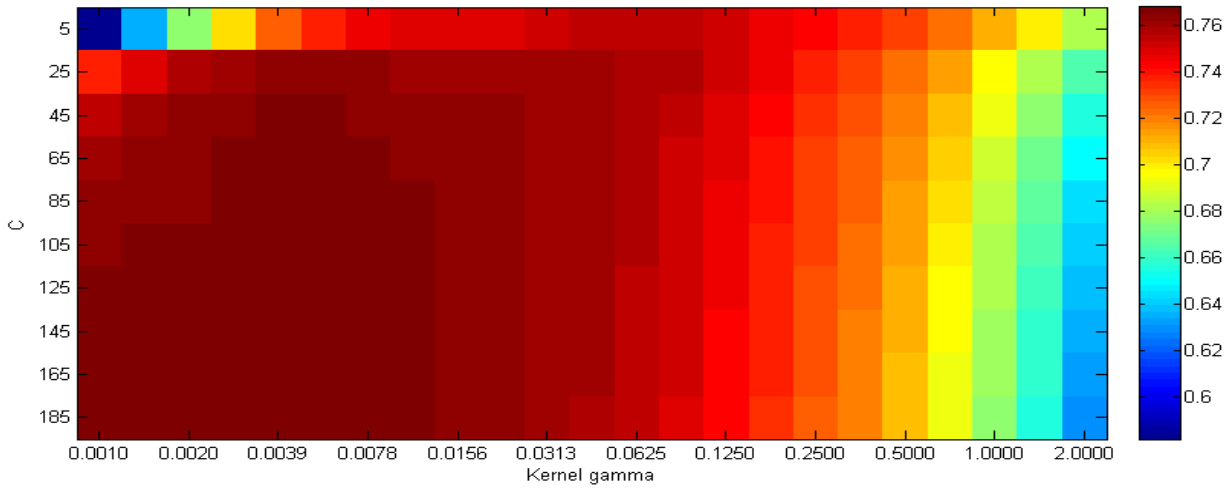

Figure 48 – Grid search over SVM parameters C and gamma. The values shown are average over the combinations of electrodes tested.

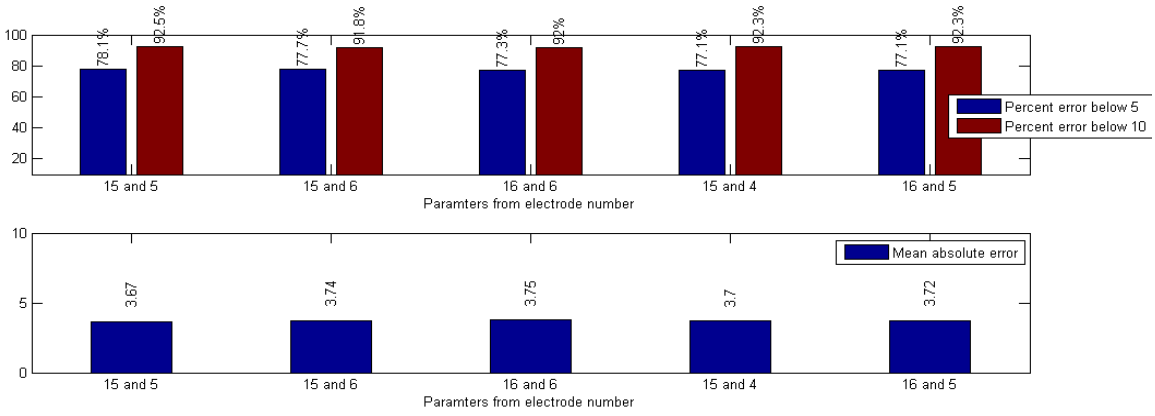

Figure 49 - Top 5 results from using two known parameters when predicting C-levels with SVM.

From Figure 49 we can see that using SVM gives about the same error as linear regression. This may imply that more complex models are not able to do any better than a linear model with two known parameters as input. We can also see that both algorithms got the highest accuracy from using the same electrodes. This could mean that what electrode is the best is independent of which algorithms we use.

Even though these values are fairly good it could be interesting to see how this method compares to other simpler method like linear interpolation (as explained in 5.4.1). If we already have measures from multiple electrodes it may actually be enough to interpolate them. Comparing the performance of these models to a simple linear interpolation will tell us if using data from previous patients actually are useful for finding the parameters of new patients. Figure 50 shows the errors using a simple linear interpolation. The errors are not

very far from what we get using other models. The best accuracy from using the database is 78.8% with linear regression, the best accuracy using linear interpolation is 72.7%. This means that 6.1% more of all patients get a good fitting using data from previous patients, compared to only interpolating the parameters.

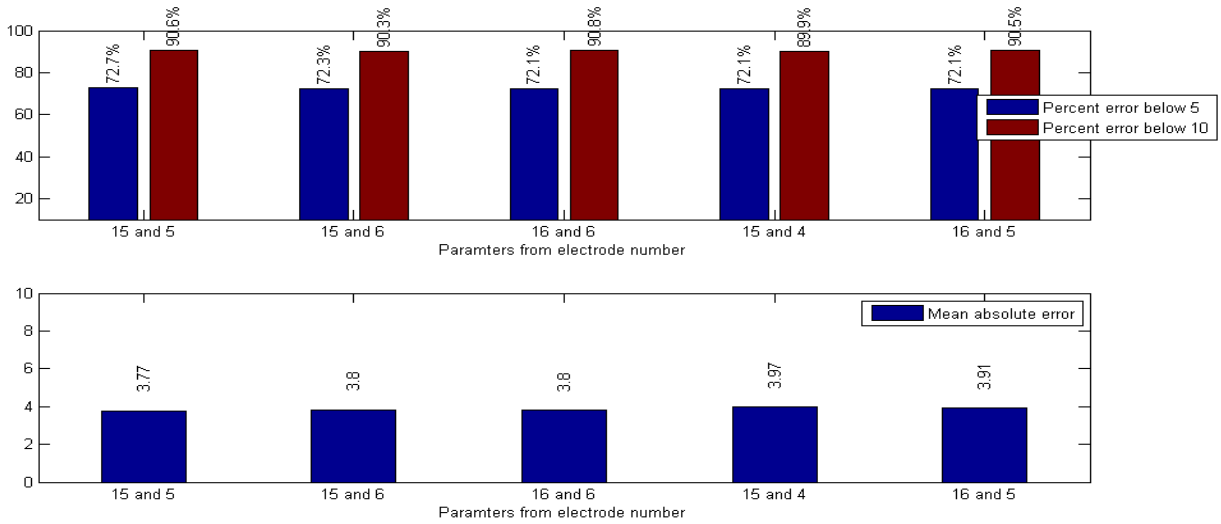

Figure 50 - Top 5 results from using two known parameters when predicting C-levels with interpolation.

Results T-level

Using linear regression to predict T-levels gives the results in Figure 51 and Figure 52. The results are similar to those we get when predicting C-levels. As we saw with C-levels, it seems that choosing electrodes that minimize the overall distance between known and unknown parameters gives the lowest error. There are however multiple choices of values that give close to the same error, but all the best combinations choose one electrode close to each end. Figure 51 illustrates what accuracy we get from each electrode combination. Using electrodes that are close to each other is not a very good option as we can see from the diagonal. The top 5 best combinations are listed in Figure 52.

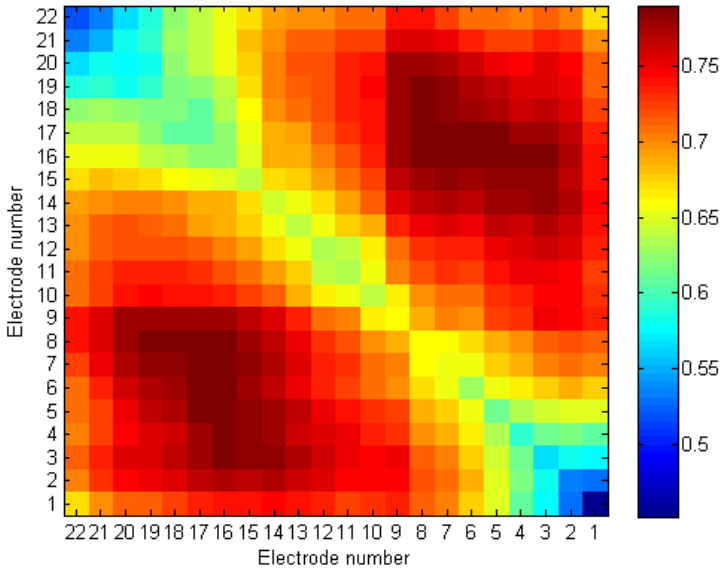

Figure 51 – Accuracy of all combinations of models using two known electrodes to predict T-levels.

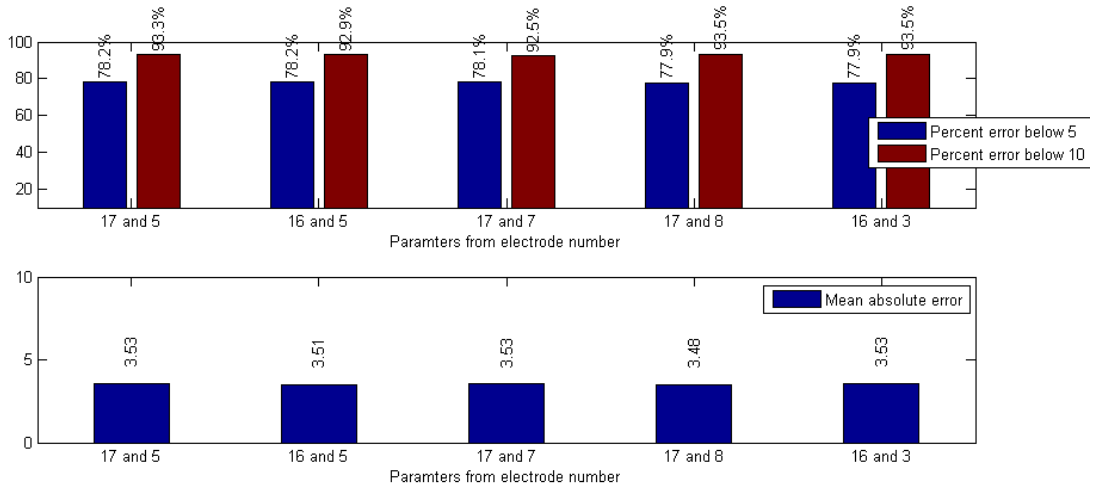

Figure 52 – Top 5 results from using two known parameters when predicting T-levels with linear regression, using two known parameters.

In the previous section a grid search to find the optimal parameters for the SVM was performed. A similar search will be done here; as before the top 20 best combinations of electrodes from linear regression will be used in the search. The result from this search can be seen in Figure 53. The result is very similar to the one for C-levels. It seems that a C above 85 and a gamma below 0.0039 gives best results. Since C is related to the complexity of the model a smaller value is often preferable if it gives the same performance. Similar a low value of gamma gives a more flat function, since each point have a larger area of impact. The results

from using SVM to predict C-level is shown in Figure 54, the parameters used where C equal to 85 and gamma equal to 0.001.

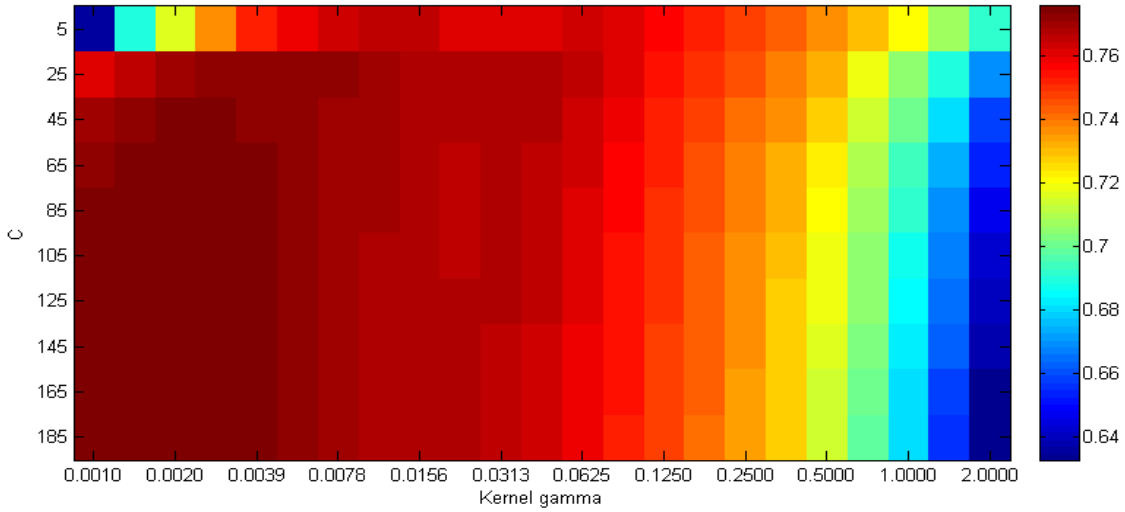

Figure 53 - Grid search over SVM parameters C and gamma. The values shown are average over the combinations of electrodes tested.

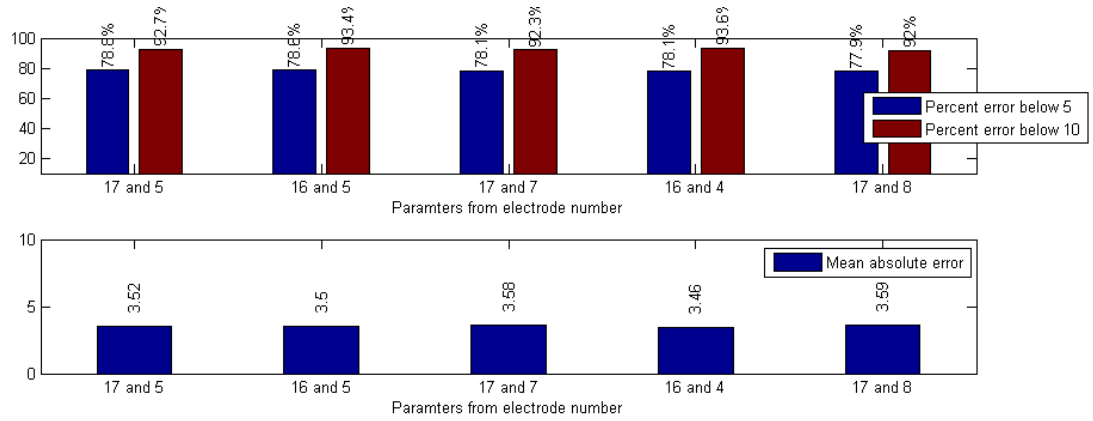

Figure 54 – Top 5 results from using two known parameters when predicting T-levels using SVM with RBF kernel, using two known parameters.

SVM and linear regression gave very similar results, the best result from linear regression where 78.2% and the best result from SVM where 78.8%. To see how good this result actually is we can compare the result to what we get from doing the same experiment using linear interpolation (see Figure 55). The best result from linear interpolation is 76%. This means that using the database give a 2.2% increase in accuracy over not using the patient database for T-levels prediction.

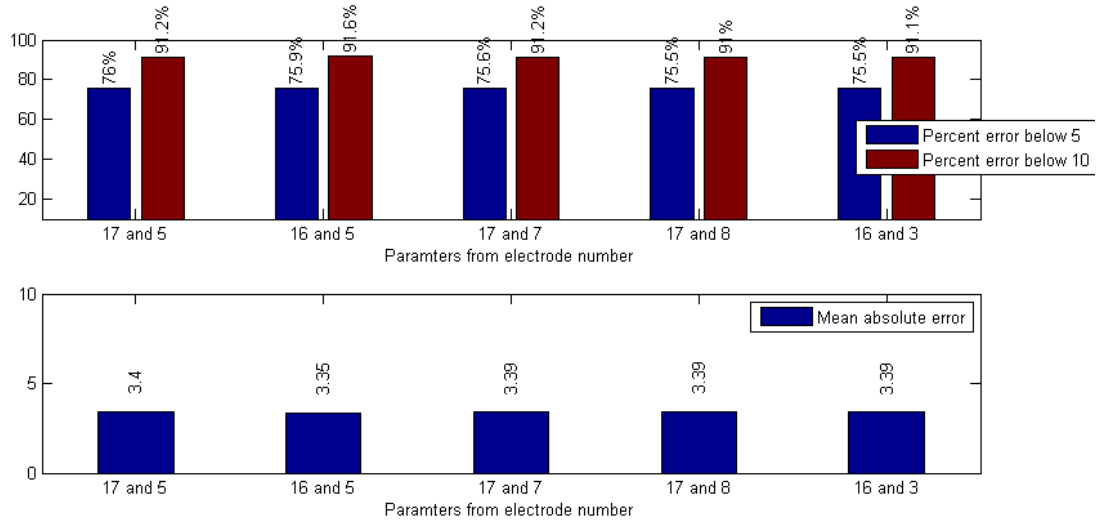

Figure 55 – Top 5 results from using two known parameters when predicting T-levels with linear interpolation, using two known parameters.

5.5 Using both objective measures and known parameters

The previous experiments have showed what accuracy we may get by using the objective measures or the known parameters as input to the model. The first experiments with only the objective measures showed an improvement of a few percent over a constant function. This means that the objective measures probably contain at least some useful information. We also saw a few percent improvement over linear interpolation when some parameters were known. It therefore seems plausible that using both objective data and known features can give an even better model. One problem is that the known parameters are very good for predicting the other parameters, while the objective measures only seem to contain a very small amount of relevant information. Section 4.3.3 discusses the implications of using features with a large amount of noise. So even though the objective measures contain some relevant information, they may actually give a negative impact when used with already known parameters.

To choose features that will be used with known parameters the, the features that performed best in the previous experiments will be used instead of doing a whole new feature selection. This is to avoid overfitting and it seems likely that the parameters that performed best in earlier experiments will also perform well now. To avoid doing another parameter search for the SVM and reduce to change of overfitting, the parameters that were chosen based on the previous searches will be used; these parameters are shown in Figure 33 and Figure 48 for C-level, Figure 32 and Figure 53 for T-level.

5.5.1 One known parameter and objective data.

The results in Figure 56 and Figure 57 show the performance of models predicting C- and T-levels, respectively. The features chosen were those that performed best in 5.3 and 5.4.3, where the best objective measures and the best known features were found. However for T-level ESRT was chosen over the combination Impd, ESRT_f1 and Impd_f1, since the performance was almost equals with ESRT alone. Even though ESRT performed slightly worse using less features are preferable, since this decrease the chance of overfitting (see 3.4.2).

It seems that using the objective measures does not decrease the error, instead it seems to increase. Using linear regression with only one known parameter and without the objective data gave an accuracy of 61.27% when predicting C-levels, while with the objective data gave an accuracy of 60.5% (see Figure 56). When predicting T-level with only one known parameter without the objective data using linear regression gave an accuracy of 65.91%, using the objective measures gave accuracy of 65.2% with linear regression (see Figure 57).

C-level

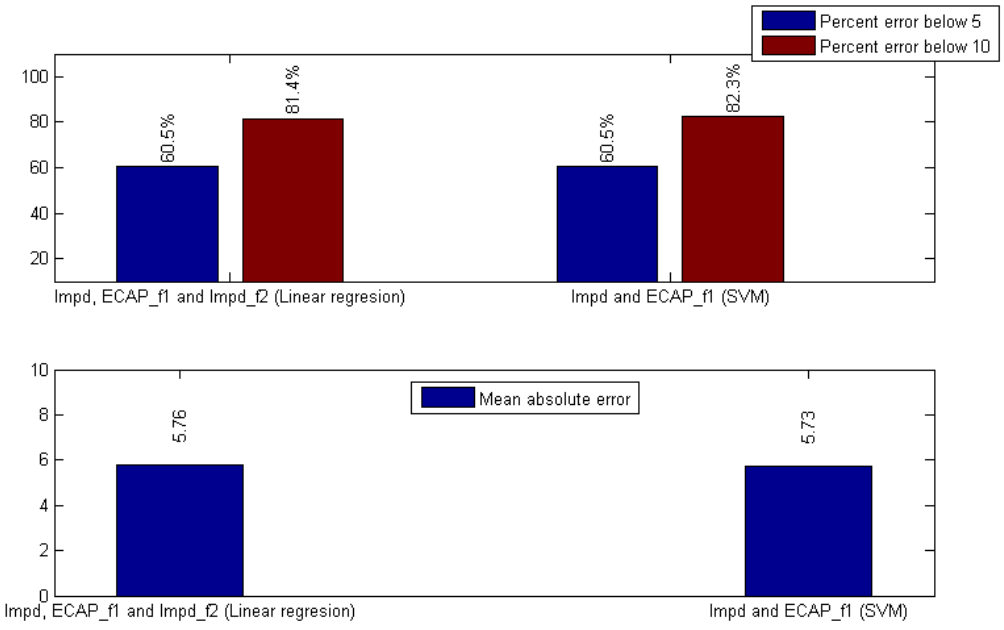

Figure 56 – Results using one known parameter to predict C-levels. The electrode chosen is the one with best result in 5.4.3, which for C-levels was electrode 15. SVM uses RBF kernel with parameters C set to 80 and gamma set to 0.001

T-level

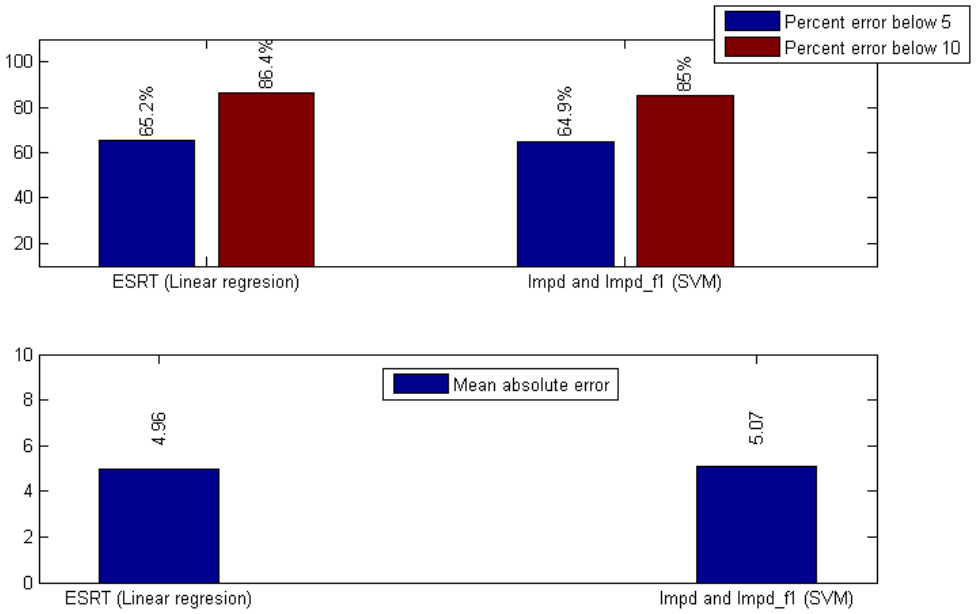

Figure 57 - Results using one known parameter to predict T-levels. The electrode chosen is the one with best result in 5.4.3, which for T-levels was electrode 9. SVM uses RBF kernel with parameters with parameters C set 80 and gamma set to 0.005

5.5.2 Two known parameters and objective data

The result in Figure 58 and Figure 59 shows the performance of models predicting C- and T-levels with two known parameters and objective data. As in the previous section features are chosen based on their individual performance in 5.3 and 5.4.3, where the best objective measures and the best known features were found. What features that were chosen are listed under each bar. As in the previous section using the objective data seems to reduce model accuracy. Using linear regression gave an accuracy of 78.3% for C-level using only the known parameters, while the best with known parameters and objective data give an accuracy of 77.7%. For T-level the accuracy dropped from 78.8% to 77.9%.

C-level

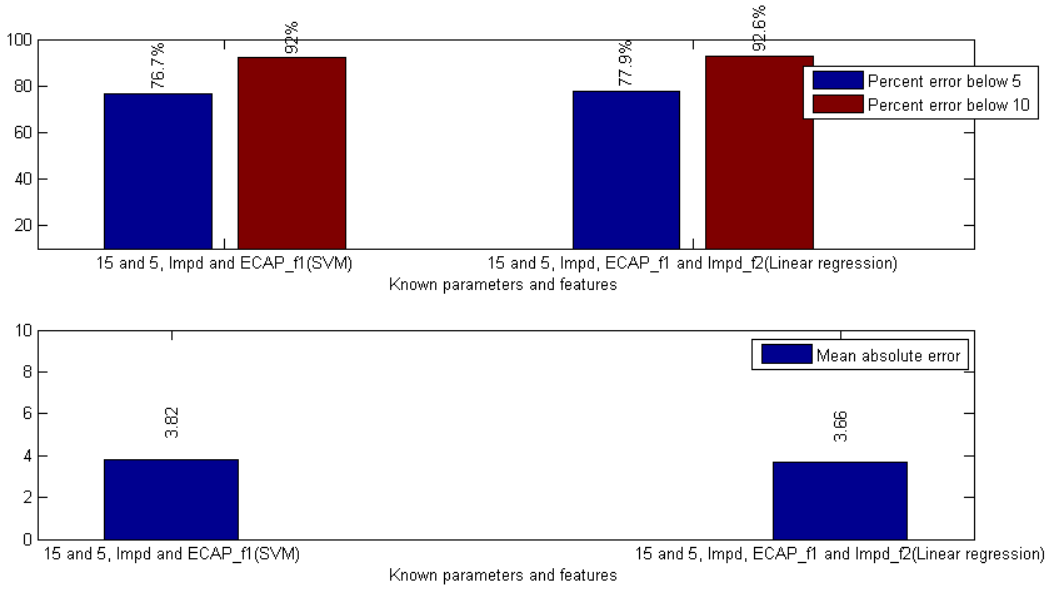

Figure 58 - Results from using both objective measures and known parameters as input to various models, when predicting C-levels. SVM uses RBF kernel with parameters with parameters C set 80 and gamma set to 0.005

T-level

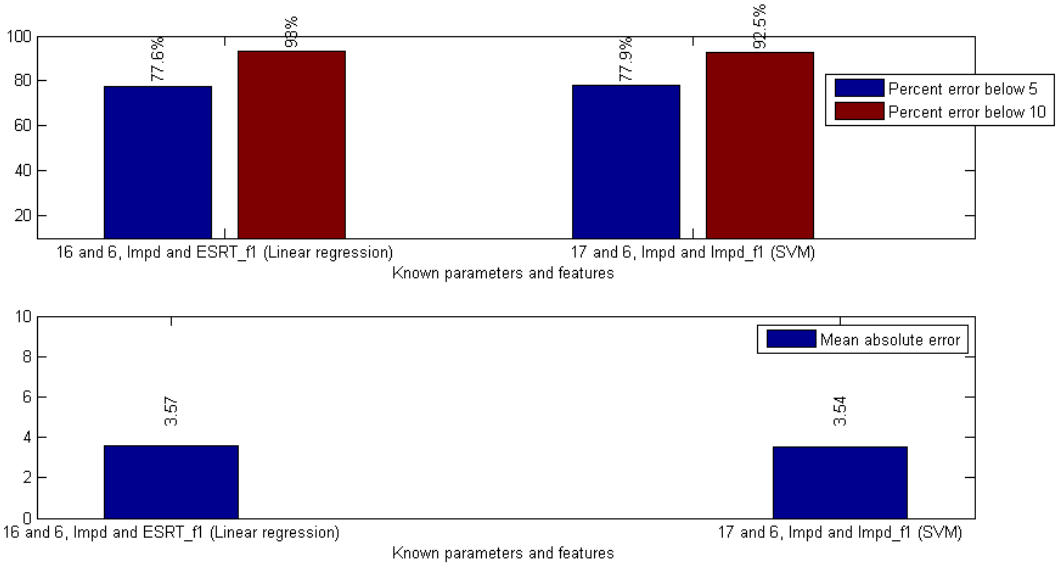

Figure 59 - Results from using both objective measures and known parameters as input to various models, when predicting T-levels. SVM uses RBF kernel with parameters with parameters C set 80 and gamma set to 0.005

5.5.3 Checking for underfitting

Section 3.4.3 discussed a method for analysing if a model is underfitting or overfitting. We can do this by plotting the cross-validation error as a function of model complexity. By doing

this it is possible see how the error changes as the complexity increase, and thereby determining if the model is overfitting or underfitting.

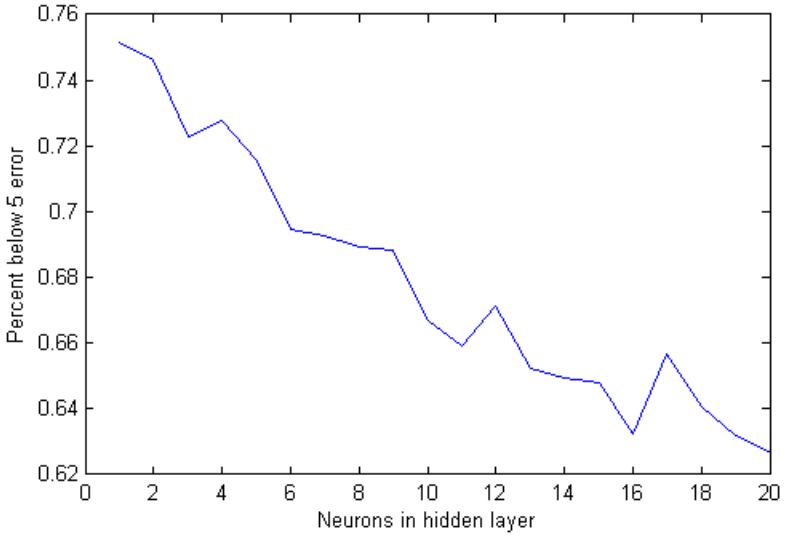

Figure 60– Percent of errors below 5 plotted as a function of number of neurons in hidden layer. The NN use the training function Levenberg-Marquardt.

From Figure 60 it is easy to see that increasing the number of neurons does not increase the accuracy of our model. The best accuracy is with only one neuron, which gives a model with very low complexity. This means that the model overfit even when the model has low complexity. It seems like whenever a non-linear model is used the model overfit, but a linear model is not able to fit the data well enough.

In the first experiments with only the only objective measures we saw that using the objective measures was better than not using any data. Which implies that the data contain some useful information, so we have to ask the question, why does not the accuracy improve when using objective data with known parameters? Intuitively it seems that adding information even though it has high uncertainty should be better than nothing. It appears to be three possible reasons that the accuracy did not improve:

1) Using the objective data and the known parameters increase the number of parameters. As discussed in 3.4.2 a large amount of parameters can have a negative impact on the model, since more features makes the model more likely to overfit. It seems quite clear that overfitting is happening in the NN when looking at Figure 60, so maybe overfitting is happening in the other models as well and thereby reducing the model accuracy.

2) Overfitting seems likely for NN and SVM, but it seems a bit unlikely that the linear model is overfitting a lot, since the error is almost equals as with only objective measures and a linear models is generally not prone to overfit. The problem may however lie in how features are weighted in linear regression. Since we are minimising the sum of squared error, using linear regression tends to follow what is called "winner takes all". Meaning that the model mainly relies on the best feature in cases where two or more features are correlated. This means that having one good feature and one a bit worse will make a linear model focus almost solely on the best feature. This could make the linear model not be able to utilize multiple known parameters.

3) There could also be a problem in how the samples are weighted. It now seems very likely that either the data is very noisy or it is necessary with some non-linear model. Whichever is the case does not really matter to a linear model, since complex data and noise is both simply data that the linear model cannot understand. If this is the case we could improve the model accuracy by using a model that can handle noise and outliers better.

5.6 Robust Linear regression and Linear SVM

The previous experiments have shown that using objective measures with known parameters gives fairly good accuracy, but not much better than using only the known parameters. This section will first investigate if a SVM with linear kernel will give better result than linear regression in 5.4. Then, see if it is possible to create a robust model that use both objective data and various numbers of known parameters using the ensemble learning method described in 3.3.4.

5.6.1 SVM with linear kernel

Linear regression and SVM with linear kernel is similar in many ways and the results are likely to be fairly equal, but it may be possible to achieve a small improvement because of how SVMs weight features and tries to minimize the model complexity (see 3.3.3). This section will show how a SVM with linear kernel performs when one and two known electrodes are available, compared to the best results from 5.4.

C-level

The results in Figure 61 shows how SVM with linear kernel perform using one and two know parameters. Using one known parameter gives very similar result as with linear regression; both have an accuracy of 61.27%. Using two electrodes gives an accuracy of 79.01% while with linear regression it was 78.3%.

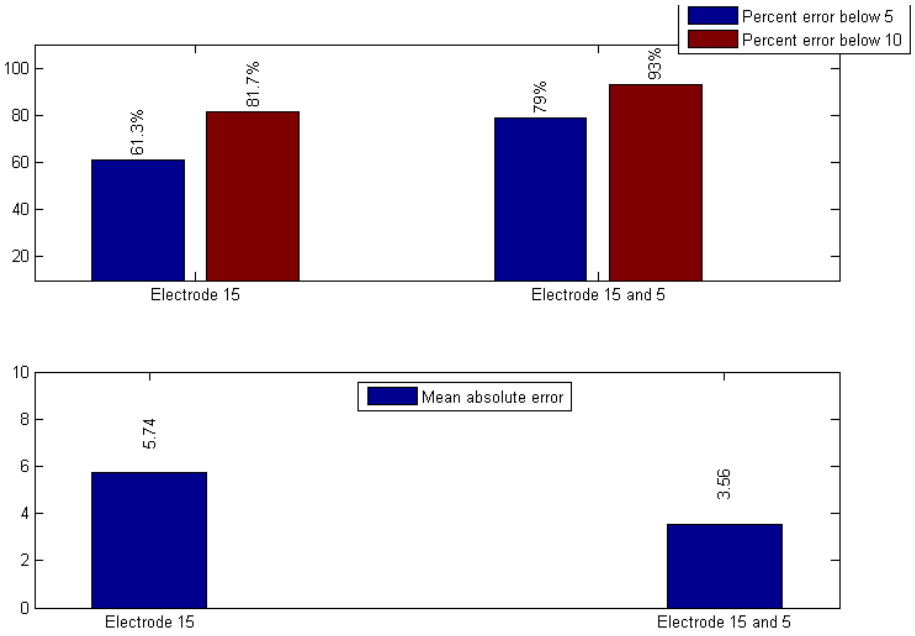

Figure 61 – Shows how SVM with Linear kernel performs with one and two known electrodes, when predicting C-levels.

T-level

Using one and two known parameters in a SVM with linear kernel to predict T-level gives the results in Figure 62. Using linear regression with known parameter from electrode 9 gave an accuracy of 65.91%, with SVM we now get 67%. Using two known electrodes an accuracy of 78.8 was achieved using SVM with RBF kernel; with linear kernel we now get 80.1% accuracy.

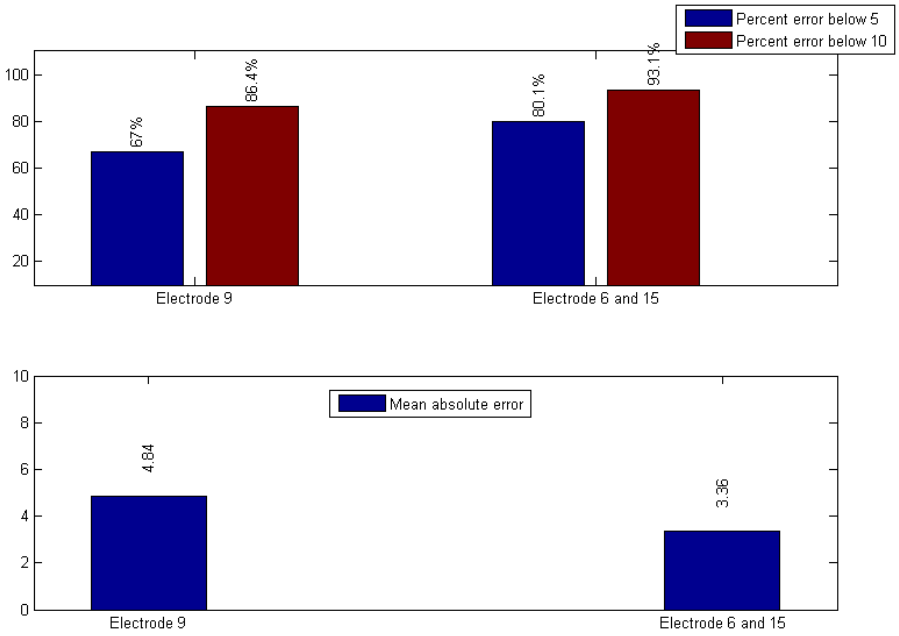

Figure 62 – Shows how SVM with Linear kernel performs with one and two known electrodes, when predicting T-levels.

5.6.2 Ensemble learning

During the experiments the performance of models using the objective data has been disappointing. However there seems fairly clear that the objective data do contain some useful information, since only using objective measures give higher accuracy than a constant function. What is also noticeable is that most of the features seem to add some information, but combining all features gives poor results. It is hard to say for sure why the performance decrease when a large number of features are used, but the likely reason is overfitting. There are many ways to deal with overfitting and a few have been tried out without great success. Another method that can be used to reduce overfitting is some sort of boosting or ensemble method (see 3.3.4). It is common when creating an ensemble model to create several models that each train on different parts of the dataset in order to create variations between the models. Since our dataset is quite small this may not be the best approach. Another way could be to train multiple models on the various combinations of features. Section 5.3 did an exhaustive search to find which objective features predict the objective measures with best accuracy. These results can be used here as well, to figure out which features should be used in the ensemble model. How many models should be trained is hard to say, but using the top ten feature combination we get all features with fairly good performance. Even if this method does not give a more accurate model, it is likely to be much more robust. Since the results have been fairly poor when using the objective data, it is probably a good idea to make sure the model is robust and not overfitting. This is especially important in practice, since a too large C-level can be uncomfortable for the patient (see 2.2.3).

Results C-level

The results in Figure 63 show how the model performs when predicting C-levels using the objective data with a various number of known parameters. When predicting with only the objective data the accuracy is quite low (14.8%), compared to when using only linear regression in 5.3, which gave an accuracy of 18.5%. The MAE is about the same, 18.9 with linear regression and 18.8 with boosting. The goal of using ensemble learning was to make something that is more robust while still not underfitting too much, which seems to have been achieved. With one and two known parameters the results now also seem slightly better than without the objective data. One known electrode gave 61.27% accuracy, when not using objective data in 5.4, where it now with boosting gives 62.6%. Two known electrodes gave 78.3% with linear regression and now give 79.2% with boosting. Not only is the accuracy a bit higher, but the model is also less likely to have overfitted, compared to some of the previous models in 5.5.

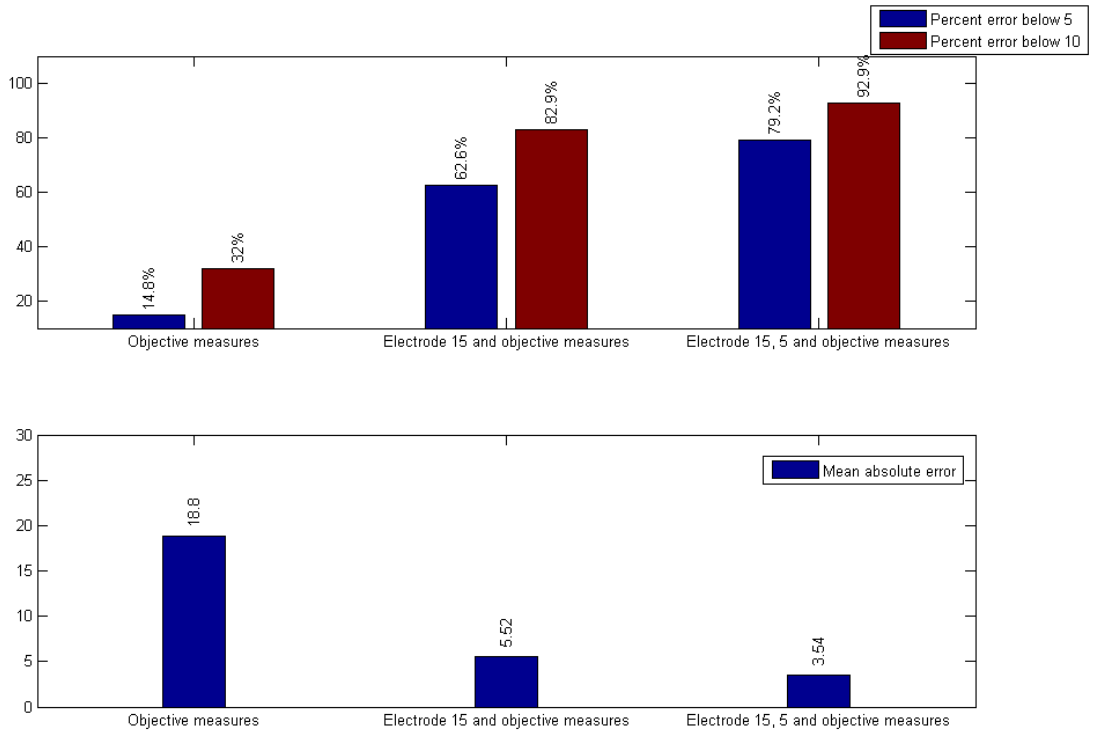

Figure 63 – Performance of how the ensemble model performs when predicting C-levels with objective measures and various number of known parameters.

Results T-level

The results in Figure 64 show how a model using the objective measure and various numbers of known parameters perform, when predicting T-levels. The results with only objective results are very similar to those in 5.3, where SVM gave an accuracy of 20%. Using one or two known T-levels with objective data does not seems to increase the model performance. Using linear regression with one known parameters gave an accuracy of 65.91%, with boosting we get 66.1%. Even though this is an improvement it seems too small to be relevant and the simpler model should probably be preferred. With two known parameters the accuracy goes from 78.2% with linear regression, to 78.9% with boosting. This indicates that using ensemble model to predict T-levels with known parameters and objective data, is almost no better than using a simple linear model with only known parameters.

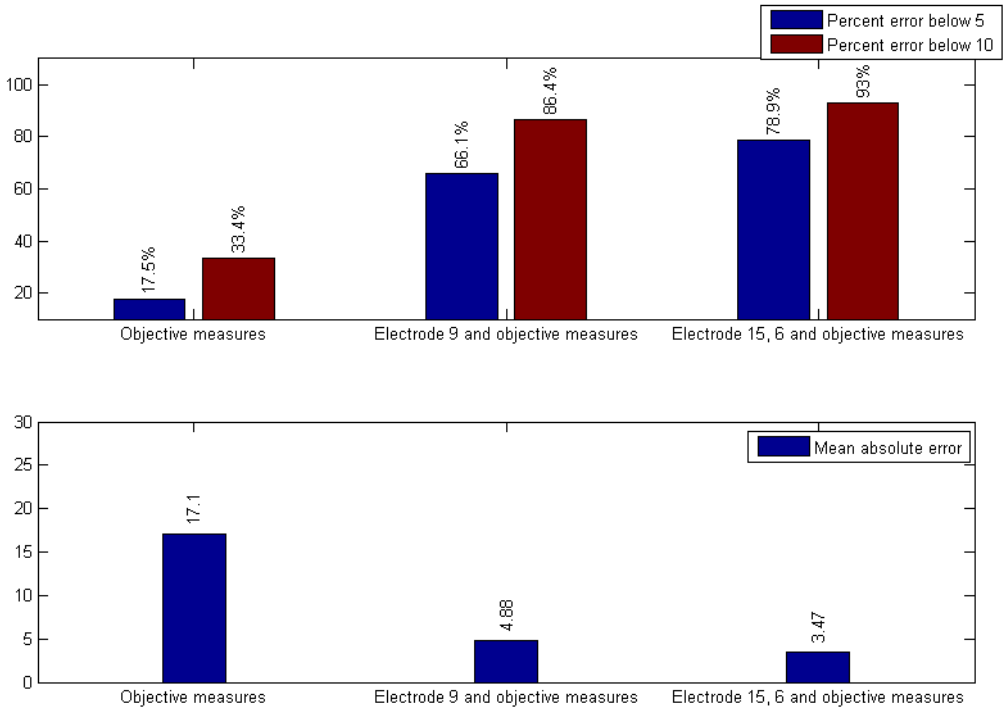

Figure 64 - Performance of how the ensemble model performs when predicting T-levels with objective measures and various number of known parameters.

5.7 Comparing model performance

During chapter 0 there have been quite a lot of various experiments, some experiments gave very poor results while others gave better. This section will try to summarize some of the most important results and discuss which model seems likely will perform best in practice. This section will not compare the exact error of the models since the values we saw in the experiments are performed with cross-validation and the results may be slightly overly positive. This can happen because of multiple iterations of changing the features and models; this was also discussed in 3.4.3. This section will rather focus on which algorithm that seems most likely to do well in practice, while still using the cross-validation performance is a guideline. A more detailed discussion of model performance will be discussed in section 5.8, where the models are tested on the test set.

5.7.1 Best model with only objective measures

The first experiments used only the objective measures to predict T- and C-levels (see 5.2) and the results showed that some objective measures were more useful than others (Figure 30 and Figure 31). ESRT and Impedance seemed in general to be the best features for both T- and C-levels. In 5.3 some alternative features and non-linear models were tested. Using these features and models seemed to improve the accuracy a bit over linear regression, but since a

large amount of parameters and features were involved in the process, it is possible that the model is overfitting to some extent. In 5.6.2 a more robust model was created using ensemble learning based on multiple linear regression models. Using this model the accuracy dropped a bit, which was somehow expected. Since the percent of errors blow 10 and MAE was still pretty good this still seems like the preferable model. The boosted model is also likely to be much more robust than the other models created and are less likely to have overfitted.

5.7.2 Best model one and two known parameters

Using one or more known objective measures did improve the results significantly in all the experiments. Which combination of electrodes gave best results was also found. T- and C-levels did not give the same electrodes, but both preferred an electrode close to the middle when using one known parameter and when using two known parameters two electrodes close to each of the ends seems preferable. The best known parameters seem to be electrode 15 when one known C-level is available (see Figure 42 and Figure 44) and electrodes 15 and 5 when two C-levels are available (see Figure 47 and Figure 52). For T-level the best known electrodes seems to be electrode 9 when one is available and 17 and 5 when two are known. Various models have been tested with these electrodes and while there have not been much difference; the linear SVM seems to have performed best (see Figure 61 and Figure 62). SVM with linear kernel is also preferable since it is a simple and robust model.

5.7.3 Best model known parameters and objective data

Using the objective data with known parameters did not give the results expected at start (5.5). It seemed that using poor features (the objective data) with very relevant data (known parameters) gave no better results than using only the known parameters (see 5.5). It seems that the objective data contains a large amount of noise or have a very complex pattern that is hard to find with a small dataset. Section 5.6.2 showed how ensemble learning can be used to make a more robust model with the objective data. Using boosting with known parameters and objective data to predict C-levels seems to slightly increase the model accuracy, over only using the known parameters. For T-levels it seems that using the objective data does not make any significant impact. It seems that the best option is to train an ensemble model when predicting C-levels, but for T-levels it should be a SVM with linear kernel and without the objective data.

5.8 Testing the model using the test set

To verify the performance of the best models they will be tested on a data set that has not been part of the earlier experiments or the data analysis. All the previous experiments have used cross-validation on the training set to estimate performance. Using a separate test set is necessary because the model may have overfitted during training and feature selection (as

explained in 3.4.4). This section will show how well the models perform on the test set and thereby give an indication for how the model may perform on new data and in practice. The errors listed are from using the models and features chosen in 5.7. Since the test set is quite small (35 samples), which may cause some inaccuracy in the error; the cross-validation error will be listed as well. If the test set error and the cross-validation error agree we may trust the result more.

5.8.1 C-level

The performance when testing the model using objective data to predict C-levels can be seen in Figure 65. The results for the test set is not all that great, but it an improvement over a constant function. The accuracy is actually higher for the test set which may indicate that the training set and the test set is a bit different, but as we can see the constant function is also higher. It seems that a model is able to utilize some of the information in the objective data.

In Figure 66 we can see the performance of the model when there is one known parameter. The performance using one known parameter and objective measures is also listed. Using a constant function on the test set with one known feature gave an accuracy of 55.9%. If the database and objective data was used the accuracy increased to 62.6% on the test set. On the validation set the results are similar with 58.9% using a constant function compared to 62.7% when using the database to train a model with objective data and one known parameter. Since both the test set and the validation set got fairly similar results it seems likely that the results are trustworthy. Using the database to train a model seems to give fairly good results over a flat/constant function.

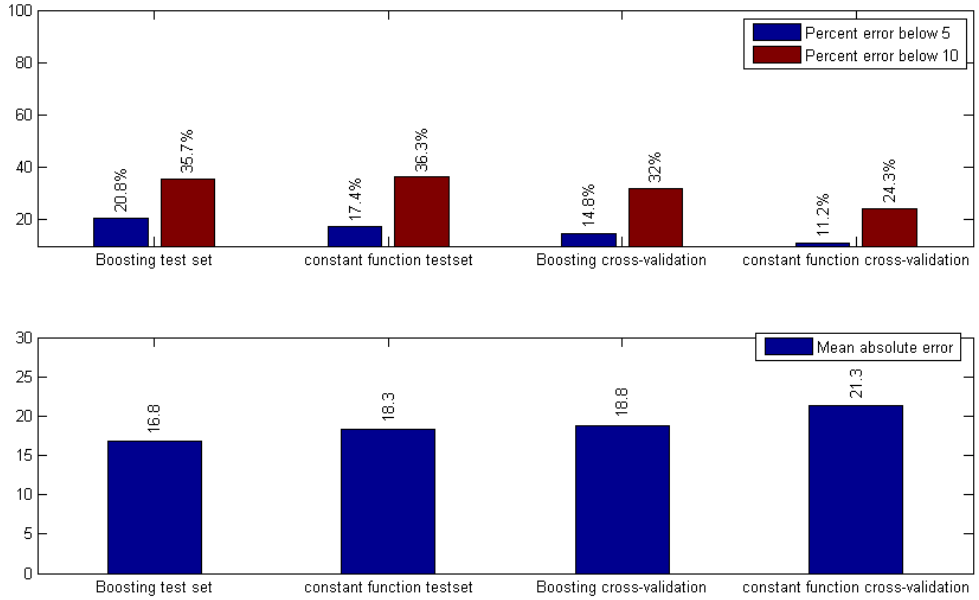

Figure 65 – Shows performance of the model when predicting C-levels on the test set and cross-validation using only the objective data. Boosting is the alternative method for fitting a robust model using linear regression as explained in 3.3.4 and 5.6.2.

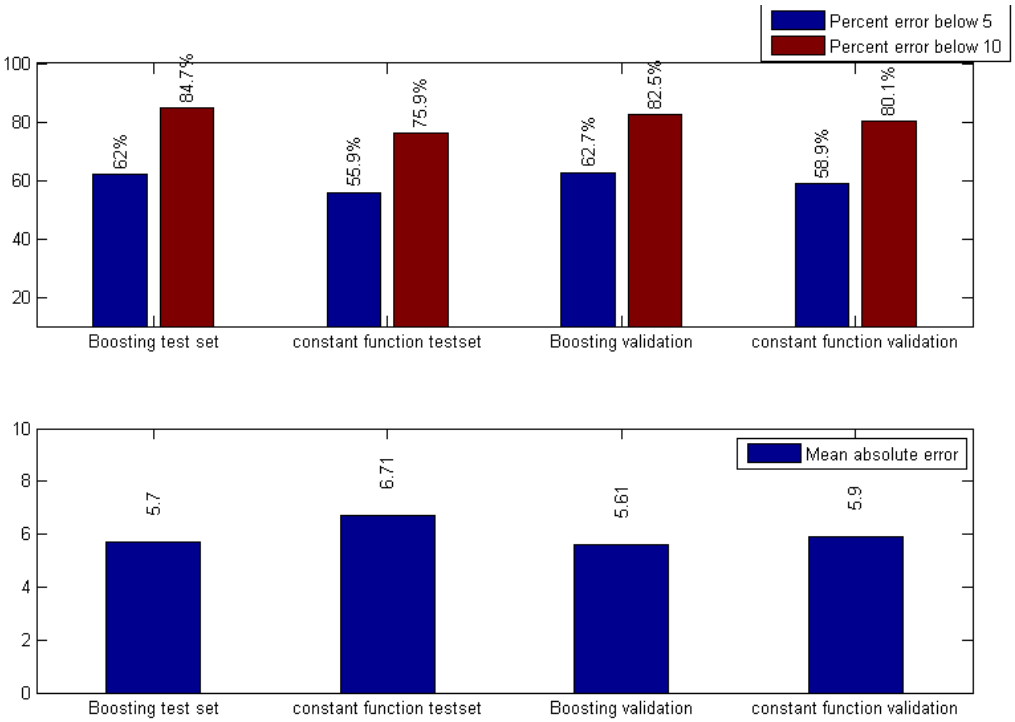

Figure 66 – Shows the various performances when using one known parameter when predicting C-levels. The performance is shown for both the validation set and the test set. Boosting is the alternative method for fitting a robust model using linear regression as explained in 3.3.4 and 5.6.2. The constant function is a function that use one known parameter and predict that value for all other electrodes on that patient (see 5.4.1).

In Figure 67 the performance when using two known parameters are listed. The test set gave an accuracy of 76.2% when using both objective measures and two known parameters. This is an improvement over using simple linear interpolation which gave an accuracy of 70.1%. For the validation set an accuracy of 79% was achieved using the database without objective data and 72.7% when using linear interpolation.

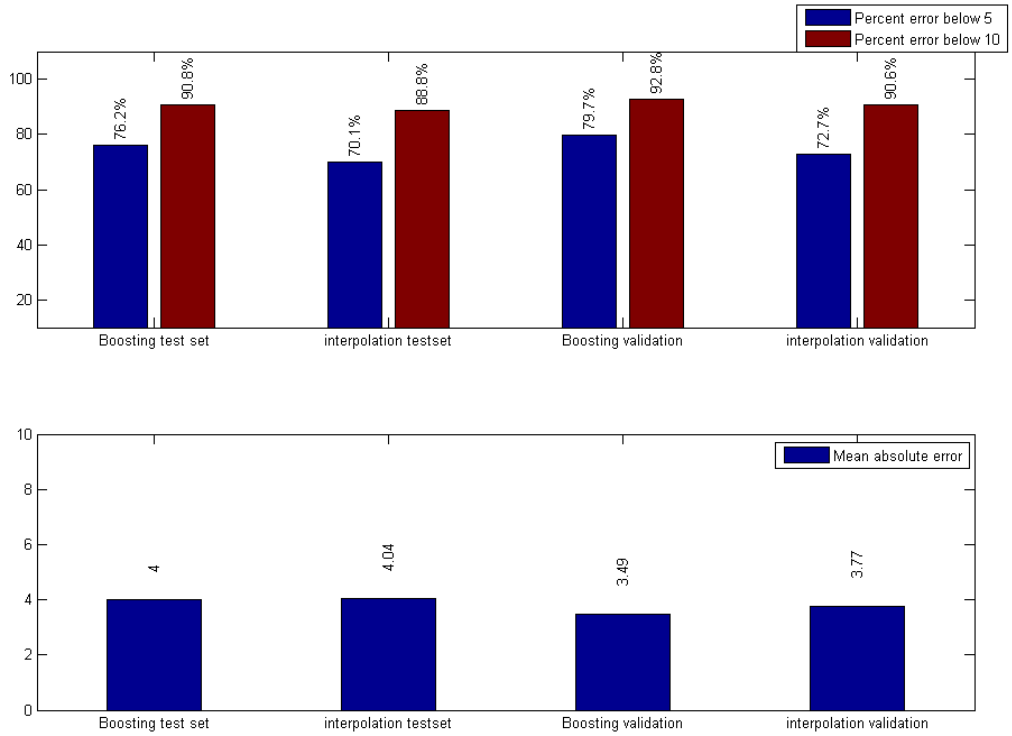

Figure 67 – Shows the various performances when using two known parameter when predicting C-levels. The performance is shown for both the validation set and the test set. Boosting is the alternative method for fitting a robust model using linear regression as explained in 3.3.4 and 5.6.2. The constant function is a function that use one known parameter and predict that value for all other electrodes on that patient (see 5.4.1).

5.8.2 T-level

Trying to predict T-level using only objective measures gave the performances shown in Figure 68. The results are fairly equal for those we saw with C-levels; it is only a few percent better to use objective measure than a constant function. The cross-validation error and the test error are fairly similar, so it seems that it is beneficial to use the objective data when there are no known parameters, but it only gives a few percent increase in accuracy.

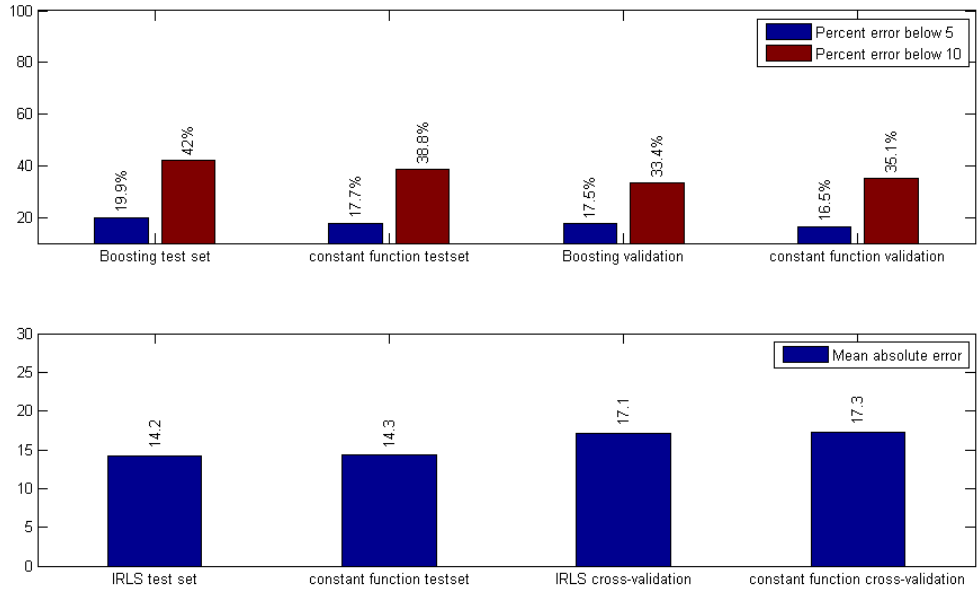

Figure 68 – Shows performance of the model on the test set and cross-validation, using only the objective data to predict T-level. Boosting is the alternative method for fitting a robust model using linear regression as explained in 3.3.4 and 5.6.2. The constant function predicts the mean of all T-levels for that electrode.

When testing the model using cross-validation, the objective data did not improve the result when used with known parameters (see 5.5 and 5.6.2). This is why a model trying to predict T-level with one known parameter and objective data was not tested. The results from testing the model with one known parameter are shown in Figure 69. Using a constant function gave an accuracy of 65% while using SVM gave 68.1%. A similar improvement can be seen on the cross-validation error where the accuracy was 67% and 62.3% for SVM and constant function, respectively.

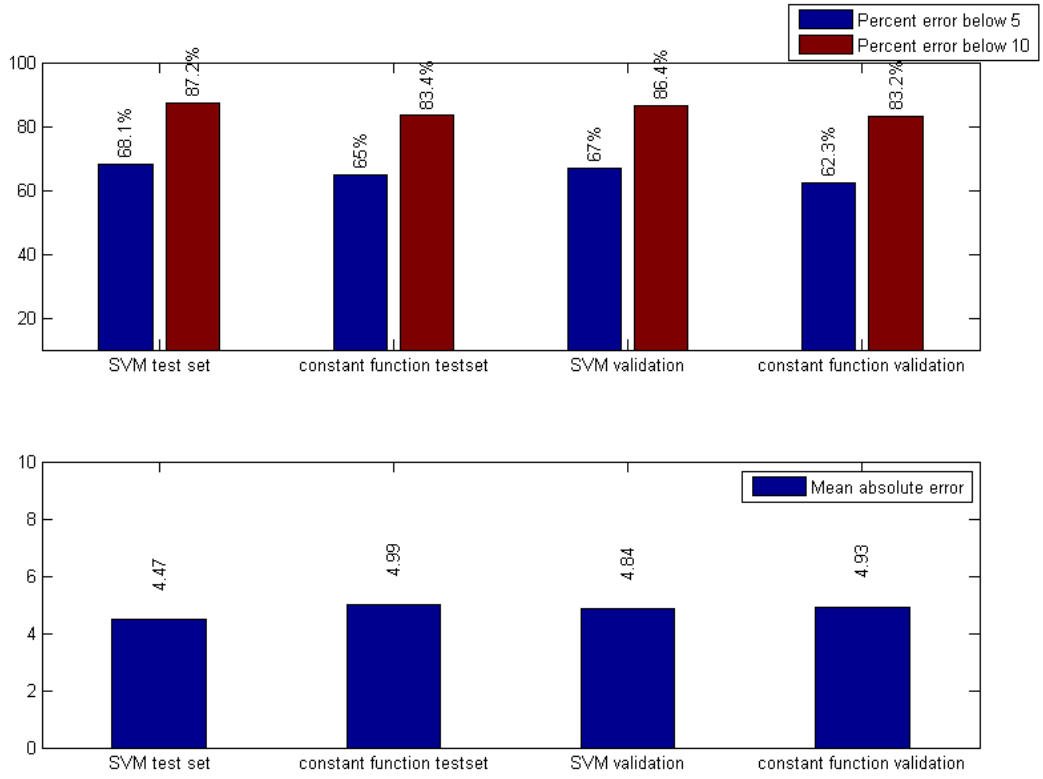

Figure 69 – Shows how SVM with linear kernel performs when predicting T-levels using one parameter as input. The constant function is a function that use one known parameter and predict that value for all other electrodes on that patient (see 5.4.1).

Same as with one known parameter the cross-validation error did not improve when using two known parameters with objective data, in comparison to only using the known parameters. The results did however improve over using linear interpolation (see Figure 70). Using SVM with two known parameters gave an accuracy of 80.4% on the test set, while interpolation gave 77.5%. The cross-validation error was very similar where the accuracy was 80% for SVM while 76% for interpolation. Comparing this to the results from C-levels in 5.8.1 it seems that predicting T-levels gives a marginally higher accuracy. However, predicting C-level seems to benefit more from using a model trained on the patient database than T-levels, since the difference between using interpolation and using the prediction models is larger for C-levels.

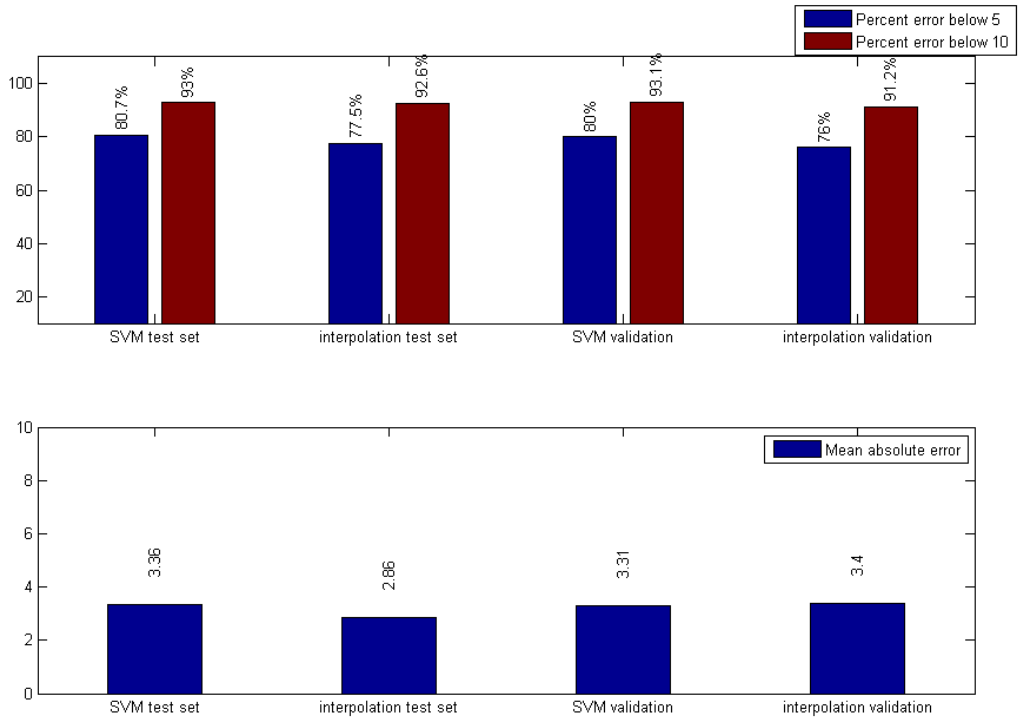

Figure 70 – Shows how SVM with linear kernel performs when predicting T-levels using two known parameters as input. The constant function is a function that use one known parameter and predict that value for all other electrodes on that patient (see 5.4.1).

6 Conclusion and further work

6.1 Conclusion

The work in this thesis first started out as trying to create a model that can predict the programming level of a Cochlear Implant speech processor, using data related to the patient. This approach is different from today's practice, where the adjustment of the CI is mainly based on feedback from the patient. The goal of this thesis has been to see how accurate predictions a machine learning model can make, that is trained on a patient database. So, to measure the performance of a model the overall number of correct predictions was used. A correct prediction was defined as +/-5CL; since patients normally have a variation in their programming levels depending on day performance (see 4.4.4). The accuracy of a model can then be estimated by looking at the total percent of correct predictions.

The various experiments showed that the objective data alone is not suited for an accurate parameter prediction. The objective data seems to either contain a large amount of noise or require a much more complex model than what can be created with such a small dataset. Even though an accurate model was not created successfully, the model created can still be useful. If a model is able to give some indication for what the parameters should be, even if it is an inaccurate estimate this can still be useful to reduce the range of parameters the audiologist needs to try out. Using the patient database a model that takes objective data as input was created. When predicting C-levels an accuracy of 19.8% could be achieved on the test set. Predicting T-levels gave an accuracy of 19.9% on the test set using the objective. Using a constant function gave 17.4% and 17.7% accuracy for C- and T-levels, respectively. This indicates that using the objective data does not give a large benefit over a constant function, but it is still an improvement.

Since the model that used only the objective data got fairly low accuracy, a more practical way to find the parameters was proposed. By using the fact that the parameters of electrodes close to each other are often highly correlated, and that in a programming session the audiologist is often able to find at least a few parameters, a model that use already known parameters can be created. This means the model could potentially be used when the audiologist is no longer able to get reliable feedback from the patient. The program would then be used with what the audiologist has found during the session, to predict the rest of the parameters. To create the models that use known parameters as input, a search was performed to find which combinations of known parameters gave most useful information about the unknown parameters. For C-levels the best results were achieved with electrode 15 when one known parameter was used and when two values were available electrode 15 and 5 gave best results. For T-levels electrode 9 gave best results when one electrode was available, when two electrodes were available electrode 17 and 5 gave best results. There are however multiple electrode pairs that gave almost equally good results, but there are some electrodes that gave

much lower accuracy than others (see Figure 46 and Figure 44). These results are quite significant, because they show that it is important which electrodes are used in interpolation or a prediction model. As discussed in 2.3.1 finding T- and C-levels for children and infants are especially challenging which makes it very important that those parameters that are found, are those which are the most useful to find the unknown parameters.

Creating a model that uses the most useful known parameters to predict other unknown parameters gave fairly good results. If one known parameter was used with objective data an accuracy of 62% was achieved for C-level on the test set. With two known parameters and objective data the accuracy increased to 76.2%. All C-levels were predicted using an ensemble of linear regression models, where the output of the ensemble was the average over the individual models. For T-levels the models were created a bit differently since using objective data with known parameters were tested, but gave no better results than using only the known parameters. Using non-linear models did not seem to improve the results either, so a SVM with linear kernel was used to predict T-level when one or two values were known. T-levels in the test set could still be predicted quite accurately with 68.1% accuracy when one parameter was known and 80.7% accuracy when two parameters were known. Moreover, since a similar accuracy was found on both the test set and during validation, it seems reasonable to assume that the models have generalized well and that these accuracies are representative for what can be expected on new patients.

As mentioned there is today no commonly used method to automatically predict T- and C-levels. So it is hard to compare how the models created in this thesis perform compared to methods used in practice today. Throughout the experiments a constant function and linear interpolation have been used as a comparison, and they do give some estimate for how much the model has learned from the data. However, without the data the linear interpolation is likely to have performed much worse without knowing which electrodes are optimal to interpolate. If we did not know which electrodes were optimal when interpolating the C-levels and the worst electrodes were chosen and interpolated (see Figure 42 and Figure 46), the average accuracy would have been 33.51% with one known parameter and 34.08% for two known parameters. If the optimal electrodes where chosen for linear interpolation, the accuracy increased to 55.9% and 70.1% for one and two known electrodes. This is still quite a bit lower than the accuracy of the model proposed, which got an accuracy of 62% and 76.2% for one and two known parameters, respectively. If the worst electrodes are chosen for T-level (see Figure 44, and Figure 51), using one known parameter gives an accuracy of 41.38% while two known parameters give 44.96%. If the two best electrodes are chosen and interpolated, the accuracy increased to 65% and 77.6% for one and two known parameters. Using the model proposed in this thesis increased the accuracy further to 68.1% and 80.7% accuracy, for one and two known parameters, respectively.

6.2 Future work

- While the experiments in this thesis showed some possible uses for the patient database to predict the parameters on new patients, it would be interesting to see how a program like this would improve the programming sessions in practice. It seems that such a program should be able to speed up the programming, but also possibly improve the final results. It would also be interesting to see how patients with CI programmed with the help of such a program would perform in a speech test, compared to other patients that were programmed without it.

- During the results I discussed various reasons that the errors were so high when using only the objective measures. Methods to test for overfitting and underfitting was performed and it seemed that linear models were not sufficient to model the data well and more complex models seemed to overfit. Using more data would make it possible to train models with higher complexity, without overfitting. This could mean that a larger database may increase the accuracy of models using only the objective data.

- In 4.1.2 I discussed various objective measures that was not available in the database but could be useful when making predictions. It would be interesting to see how a prediction model would perform if more relevant data was available.

- Throughout the experiments the error measure has been the mean overall electrodes, but the individual performance of each electrode have not been analysed. It may be smart to handle each electrode more separately and that features should be chosen for specific electrodes. This also pose the question why some electrodes would act differently, as discussed previously age can be one cause, electrode placement can be another. It would be interesting to see if it is possible to find some data that could explain some of these variations such that they could be used in a model.

References

1. WHO. *Deafness and hearing loss*. 2014 October 11, 2014]; Available from: http://www.who.int/mediacentre/factsheets/fs300/en/.
2. Tomblin, J.B., et al., *The effect of age at cochlear implant initial stimulation on expressive language growth in infants and toddlers.* Journal of Speech, Language, and Hearing Research, 2005. **48**(4): p. 853-867.
3. Thekkek, N. and R. Richards-Kortum, *Optical imaging for cervical cancer detection: solutions for a continuing global problem.* Nature Reviews Cancer, 2008. **8**(9): p. 725-731.
4. Tan, A.C. and D. Gilbert, *Ensemble machine learning on gene expression data for cancer classification.* 2003.
5. Charasse, B., et al., *Automatic analysis of auditory nerve electrically evoked compound action potential with an artificial neural network.* Artificial Intelligence in Medicine, 2004. **31**(3): p. 221-229.
6. Garg, A.X., et al., *Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: a systematic review.* Jama, 2005. **293**(10): p. 1223-1238.
7. Martin, F.N. and J.G. Clark, *Introduction to Audiology, 11/E*. 2011.
8. Pang, X. and W. Peake, *How do contractions of the stapedius muscle alter the acoustic properties of the ear?*, in *Peripheral auditory mechanisms*. 1986, Springer. p. 36-43.
9. Choi, C.T. and Y.-H. Lee, *A Review of Stimulating Strategies for Cochlear Implants.*
10. Purves, D., et al., *Two Kinds of Hair Cells in the Cochlea.* 2001.
11. Davis, H., *An active process in cochlear mechanics.* Hearing research, 1983. **9**(1): p. 79-90.
12. Cooper, H. and L. Craddock, *Cochlear implants: A practical guide*. 2006: John Wiley & Sons.
13. Van Den Abbeele, T., et al., *Multicentre investigation on electrically evoked compound action potential and stapedius reflex: how do these objective measures relate to implant programming parameters?* Cochlear Implants Int, 2012. **13**(1): p. 26-34.
14. Kral, A. and A. Sharma, *Developmental neuroplasticity after cochlear implantation.* Trends in neurosciences, 2012. **35**(2): p. 111-122.
15. Sharma, A., M.F. Dorman, and A.J. Spahr, *A sensitive period for the development of the central auditory system in children with cochlear implants: implications for age of implantation.* Ear and hearing, 2002. **23**(6): p. 532-539.
16. Bonnet, R.M., et al., *Effects of pulse width, pulse rate and paired electrode stimulation on psychophysical measures of dynamic range and speech recognition in cochlear implants.* Ear and hearing, 2012. **33**(4): p. 489-496.
17. Allum, J.H., R. Greisiger, and R. Probst, *Relationship of intraoperative electrically evoked stapedius reflex thresholds to maximum comfortable loudness levels of children with cochlear implants.* Int J Audiol, 2002. **41**(2): p. 93-9.
18. Polak, M., A. Hodges, and T. Balkany, *ECAP, ESR and subjective levels for two different Nucleus 24 electrode arrays.* Otology & Neurotology, 2005. **26**(4): p. 639-645.
19. Kim, J.R., et al., *The relationship between electrically evoked compound action potential and speech perception: a study in cochlear implant users with short electrode array.* Otol Neurotol, 2010. **31**(7): p. 1041-8.

20. Cosetti, M.K., et al., *Intraoperative neural response telemetry as a predictor of performance.* Otol Neurotol, 2010. **31**(7): p. 1095-9.
21. Potts, L.G., et al., *Relation between neural response telemetry thresholds, T- and C-levels, and loudness judgments in 12 adult nucleus 24 cochlear implant recipients.* Ear Hear, 2007. **28**(4): p. 495-511.
22. Muhaimeed, H.A., et al., *Correlation between NRT measurement level and behavioral levels in pediatrics cochlear implant patients.* Int J Pediatr Otorhinolaryngol, 2010. **74**(4): p. 356-60.
23. Clark, G.M., et al., *Cochlear implantation: osteoneogenesis, electrode-tissue impedance, and residual hearing.* Scientific publications, vol. 8, 1994-1995, no. 720, 1995.
24. Tipping, M.E., *Bayesian inference: An introduction to principles and practice in machine learning*, in *Advanced lectures on machine Learning*. 2004, Springer. p. 41-62.
25. Hornik, K., M. Stinchcombe, and H. White, *Multilayer feedforward networks are universal approximators.* Neural networks, 1989. **2**(5): p. 359-366.
26. Caudill, M. and C. Butler, *Understanding neural networks: computer explorations: a workbook in two volumes with software for the macintosh and pc compatibles*. 1994: MIT press.
27. Battiti, R., *First-and second-order methods for learning: between steepest descent and Newton's method.* Neural computation, 1992. **4**(2): p. 141-166.
28. Charalambous, C. *Conjugate gradient algorithm for efficient training of artificial neural networks*. in *Circuits, Devices and Systems, IEE Proceedings G*. 1992. IET.
29. Leung, F.H.-F., et al., *Tuning of the structure and parameters of a neural network using an improved genetic algorithm.* Neural Networks, IEEE Transactions on, 2003. **14**(1): p. 79-88.
30. Mukherjee, I. and S. Routroy, *Comparing the performance of neural networks developed by using Levenberg–Marquardt and Quasi-Newton with the gradient descent algorithm for modelling a multiple response grinding process.* Expert Systems with Applications, 2012. **39**(3): p. 2397-2407.
31. Marquardt, D.W., *An algorithm for least-squares estimation of nonlinear parameters.* Journal of the Society for Industrial & Applied Mathematics, 1963. **11**(2): p. 431-441.
32. Suratgar, A.A., M.B. Tavakoli, and A. Hoseinabadi, *Modified Levenberg-Marquardt method for neural networks training.* World Acad Sci Eng Technol, 2005. **6**: p. 46-48.
33. Yu, H. and B.M. Wilamowski, *Levenberg-marquardt training.* The Industrial Electronics Handbook, 2011. **5**: p. 1-15.
34. Boser, B.E., I.M. Guyon, and V.N. Vapnik. *A training algorithm for optimal margin classifiers*. in *Proceedings of the fifth annual workshop on Computational learning theory*. 1992. ACM.
35. Burges, C.J., *A tutorial on support vector machines for pattern recognition.* Data mining and knowledge discovery, 1998. **2**(2): p. 121-167.
36. Sanchez, A. and V. David, *Advanced support vector machines and kernel methods.* Neurocomputing, 2003. **55**(1): p. 5-20.
37. Schapire, R.E., *The boosting approach to machine learning: An overview*, in *Nonlinear estimation and classification*. 2003, Springer. p. 149-171.
38. Botros, A., B. van Dijk, and M. Killian, *AutoNRT™: An automated system that measures ECAP thresholds with the Nucleus< sup>®</sup> Freedom™ cochlear implant via machine intelligence.* Artificial Intelligence in Medicine, 2007. **40**(1): p. 15-28.

39. Hawkins, D.M., *The problem of overfitting.* Journal of chemical information and computer sciences, 2004. **44**(1): p. 1-12.
40. MacKay, D.J., *Probable networks and plausible predictions-a review of practical Bayesian methods for supervised neural networks.* Network: Computation in Neural Systems, 1995. **6**(3): p. 469-505.
41. Domingos, P., *The role of Occam's razor in knowledge discovery.* Data mining and knowledge discovery, 1999. **3**(4): p. 409-425.
42. Banko, M. and E. Brill. *Scaling to very very large corpora for natural language disambiguation*. in *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*. 2001. Association for Computational Linguistics.
43. Shepherd, R.K., L.A. Roberts, and A.G. Paolini, *Long-term sensorineural hearing loss induces functional changes in the rat auditory nerve.* European Journal of Neuroscience, 2004. **20**(11): p. 3131-3140.
44. Nadol Jr, J., Y. Young, and R. Glynn, *Survival of spiral ganglion cells in profound sensorineural hearing loss: implications for cochlear implantation.* The Annals of otology, rhinology, and laryngology, 1989. **98**(6): p. 411.
45. Janeschik, S., et al., *Influence of etiologic factors on speech perception of cochlear-implanted children.* Cochlear Implants Int, 2013.
46. Wolpert, D.H. and W.G. Macready, *No free lunch theorems for optimization.* Evolutionary Computation, IEEE Transactions on, 1997. **1**(1): p. 67-82.
47. Caruana, R. and A. Niculescu-Mizil. *An empirical comparison of supervised learning algorithms*. in *Proceedings of the 23rd international conference on Machine learning*. 2006. ACM.
48. Caruana, R., N. Karampatziakis, and A. Yessenalina. *An empirical evaluation of supervised learning in high dimensions*. in *Proceedings of the 25th international conference on Machine learning*. 2008. ACM.
49. Sterne, J.A.C., et al., *Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls.* BMJ, 2009. **338**.
50. Hamel, P. and D. Eck. *Learning Features from Music Audio with Deep Belief Networks*. in *ISMIR*. 2010. Utrecht, The Netherlands.
51. Koren, Y., *The bellkor solution to the netflix grand prize.* Netflix prize documentation, 2009. **81**.
52. Guyon, I. and A. Elisseeff, *An introduction to variable and feature selection.* The Journal of Machine Learning Research, 2003. **3**: p. 1157-1182.
53. Chang, C.-C. and C.-J. Lin, *LIBSVM: a library for support vector machines.* ACM Transactions on Intelligent Systems and Technology (TIST), 2011. **2**(3): p. 27.